Crowd Simulation

Jungdam Won

Computer Science & Engineering

Seoul National Univ.

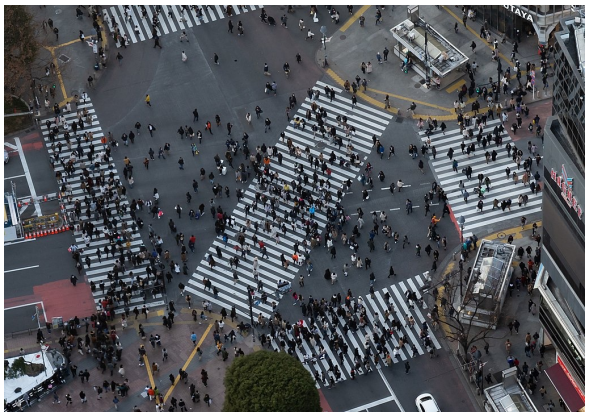
Agenda

- What is crowd simulation?
  - Definition
  - Applications

- Approaches
  - Microscopic models
    - Rule-based model (Boids)
    - Force-based model (Social force)
    - Velocity-based model (Velocity obstacles)
  - Macroscopic models
    - Continuum model

What is Crowd?

- Crowd is a group of individuals who ***share similar information in the same environment*** alone or in a group




https://www.koreatimes.co.kr/www/sports/2018/02/661_235820.html
https://en.wikipedia.org/wiki/Shibuya_Crossing

What is Crowd Simulation?

- Crowd simulation is the process of simulating the movement of a large number of entities or characters (i.e., ***crowd***) - wiki -

Applications

- VFX (Movie Making or Computer Game)
  - E.g., in movie making, let's say you are planning to shoot a scene where 10,000 people approximately are required
    - 10,000 x $200/day x 3 days = $6,000,000
  - In old days (even up to nowadays), there have been companies which implement crowd simulation in reality

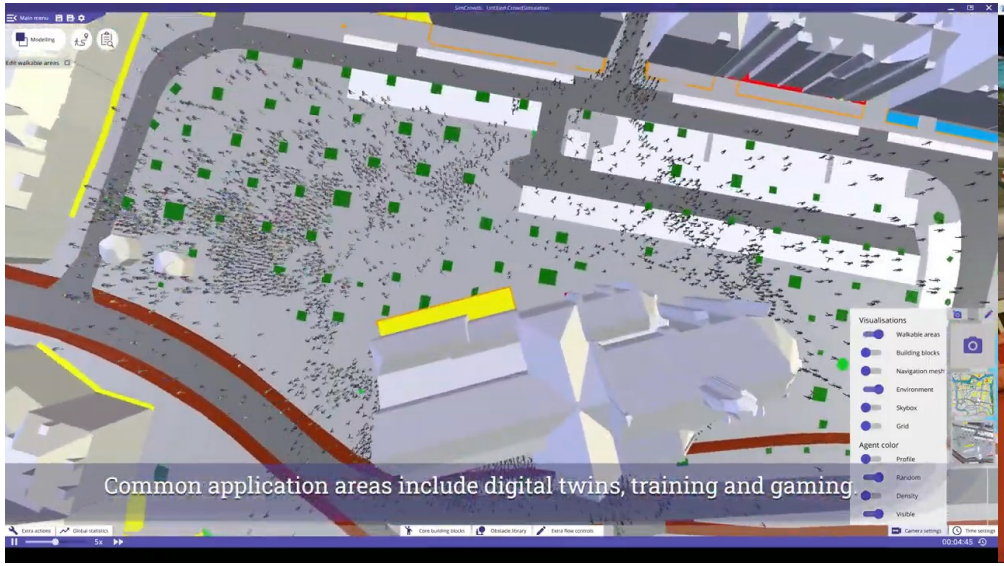
https://www.facebook.com/TheInflatableCrowd/

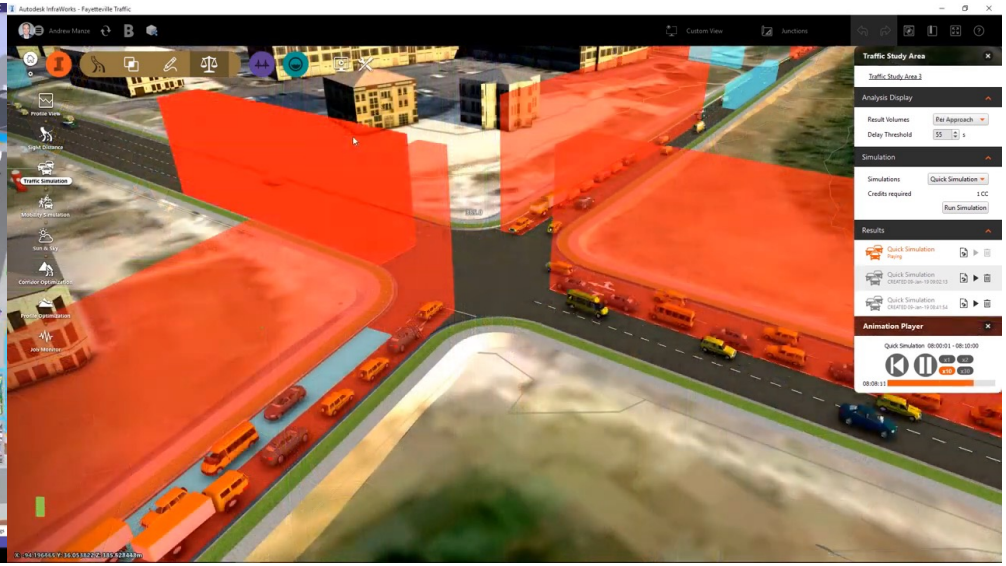
Applications

https://youtu.be/Ft1LeQiwDj8

Applications

- Urban planning
  - Pedestrian behavior in emergency and evacuation
  - Traffic simulation





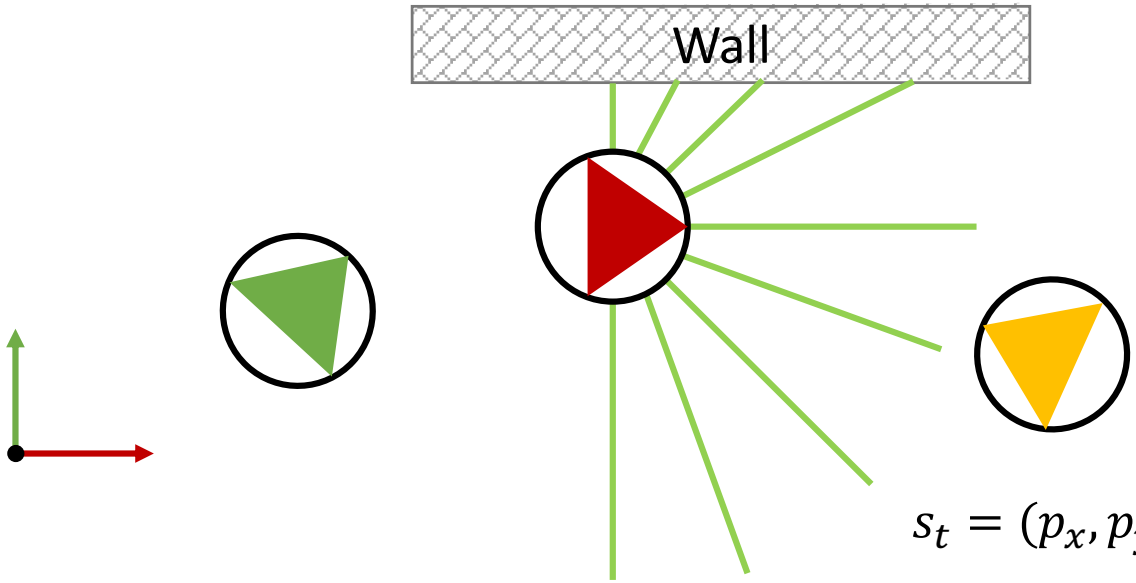
https://www.youtube.com/watch?v=zvZ5gm5YF
https://www.youtube.com/watch?v=BDKcWkggAaA

Types of Crowd Simulation

- Tiling (Static)
  - Using crowd to fill the empty background
  - The locations of agents often do not change, so their visuals and animations are more important
  - This topic will not be covered in this lecture, refer to the link below for more information
    - https://www.youtube.com/watch?v=hqIaPkTsGyA
- Navigation (Dynamic)
  - Modeling navigational behaviors (moving toward goals or wandering around)
  - Efficient navigation with collision avoidance is more important than other visual factors
  - In this lecture, we will focus more on this

Agent (Individual) Model

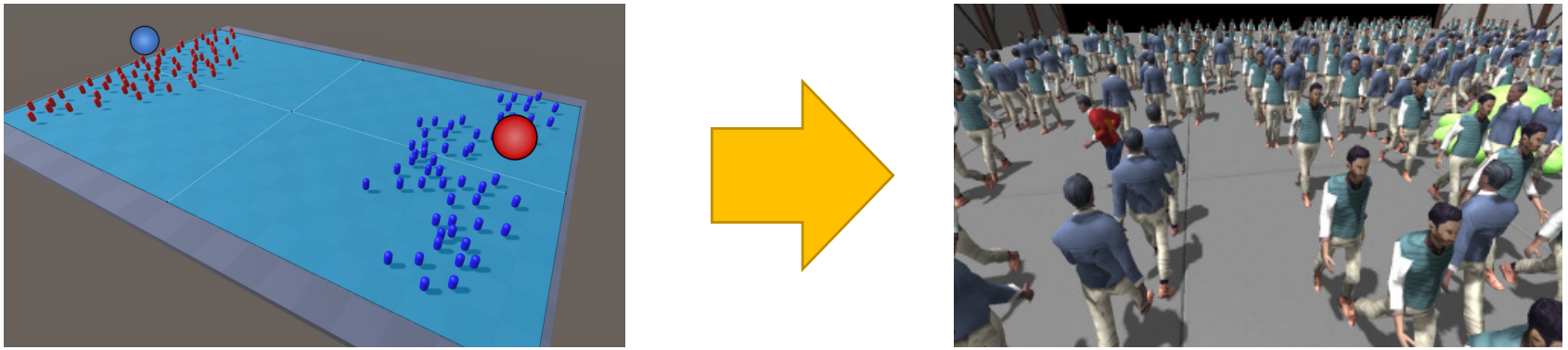
- Each agent is approximated by a 2D disk
  - ***Radius*** refers to the body size (or comfort zone) of the agent
  - The state may include ***position, velocity, orientation,*** and ***observation of surrounding environments***



$$s_t = (p_x, p_y, v_x, v_y, \theta, \omega, l_1, \cdots, l_K, \cdots)$$

Agent (Individual) Model

- Once the behaviors are simulated in 2D, full-body 3D animations can be integrated later by applying the generated translations and rotations in 2D into 3D characters



https://www.oreilly.com/library/view/unity-2017-game/9781788477901/09fc5a1b-d10b-44a3-88c9-d3be9a6a51c3.xhtml
https://gamma.cs.unc.edu/research/crowds/

Approaches

- ***Microscopic*** (bottom-up) models
  - Every agent makes a decision individually based on their own state and surroundings (neighbor and obstacles)
  - This resembles what real humans do
- ***Macroscopic*** models
  - There often exists a single decision maker which knows all information of the entire system.
  - The decision maker makes all decisions for every agents so that such decisions are optimal from the overall system point of view even if some decisions might be optimal for some agents
- ***Mesoscopic*** models
  - It models group (a few agents) behaviors
  - It will not be covered in this lecture

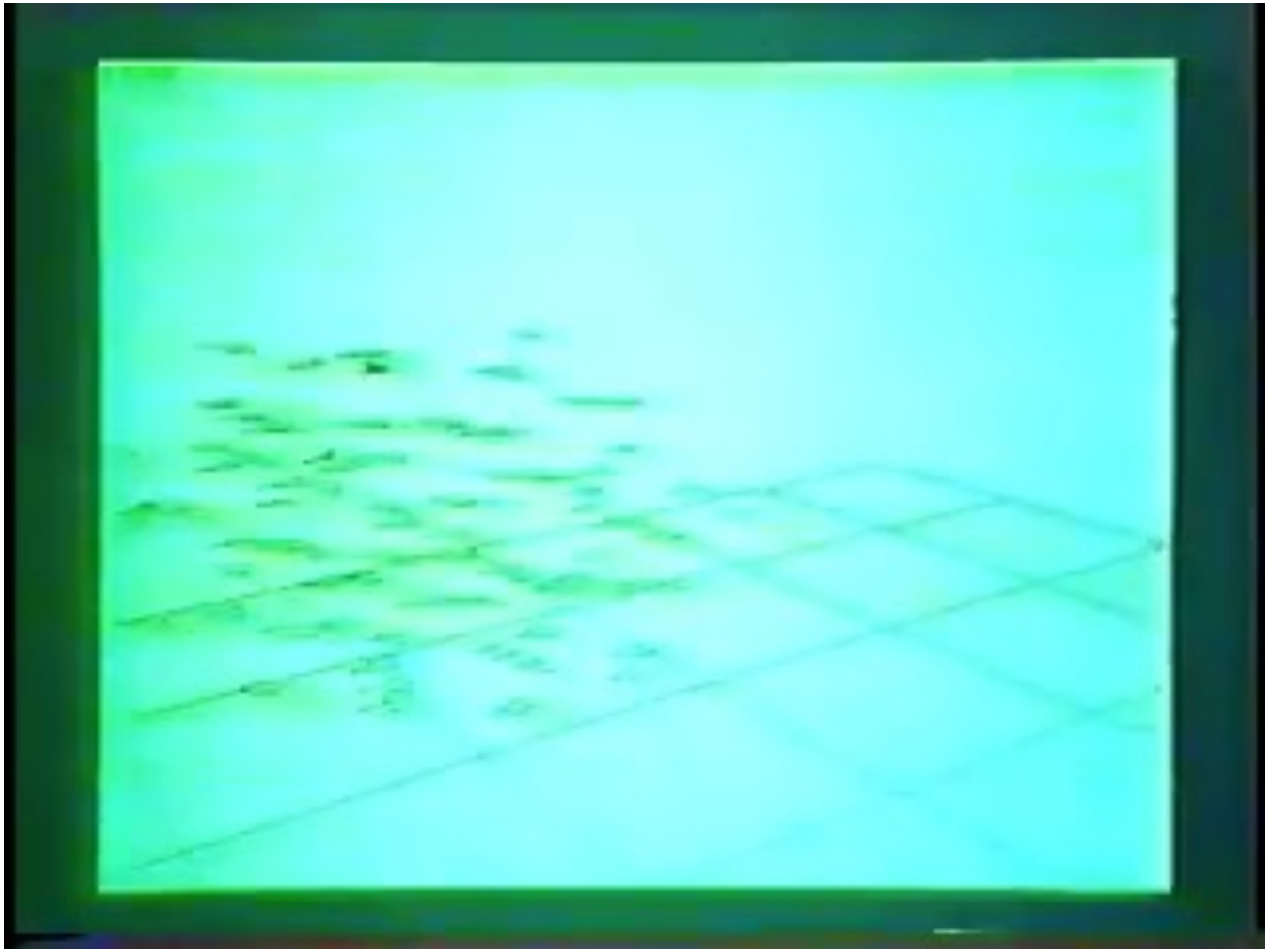
Boids (Bird-oid) Model

- The earliest crowd simulation system developed by ***Craig Reynolds*** in 1987

- Developed to simulate animal group behaviors such as flocks, herds, and schools of fishes

- It first demonstrated that complex and chaotic group behaviors can be made by a few simple ***rules***

Boids Model (Original Video)



Boids Model (by Others)

https://www.youtube.com/watch?v=Yqq5_NW-YZs

Boids Model

- Many rule-based algorithms including ***boids*** model share similar code structure like below

Initialize states
Until max-step reached
    For each boid b
        $N_b = \mathrm{FindNeighbors}(b)$
        For each rule k
            $\mathbf{v}_i = \mathbf{v}_i + \omega_{\mathrm{k}} \cdot \mathrm{rule}_k(b, N_b)$
        $\mathbf{p}_i = \mathbf{p}_i + \Delta t \cdot \mathbf{v}_i$
        Align orientation to the direction of $\mathbf{v}_i$

Rule1: Separation

- Each boid will get uncomfortable when other boids get too close, and it will try to avoid them

Procedure $\text{Rule}_{\text{sep}}(b_i, N_{b_i})$**:**

$\mathbf{v} = \mathbf{0}$

For boid b_k in N_{b_i}

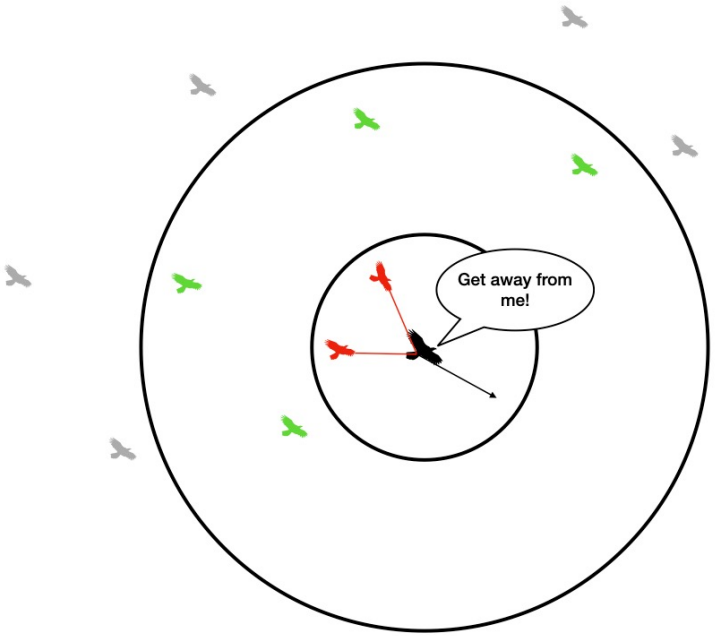
$\mathbf{d} = \mathbf{p}_{\text{i}} - \mathbf{p}_{\text{k}}$

If $|\mathbf{d}| < \text{protected\_range}$

$\mathbf{v} = \mathbf{v} + \mathbf{d}$

$\mathbf{v} = \text{avoid}_{\text{factor}} \cdot \mathbf{v}$

Return $\mathbf{v}$



https://vanhunteradams.com/Pico/Animal_Movement/Boids-algorithm.html

Rule2: Alignment

- Each boid attempts to match the velocity of other boids inside its visible range

Procedure $\text{Rule}_{\text{ali}}(b_i, N_{b_i})$:

$\mathbf{v} = \mathbf{0}$

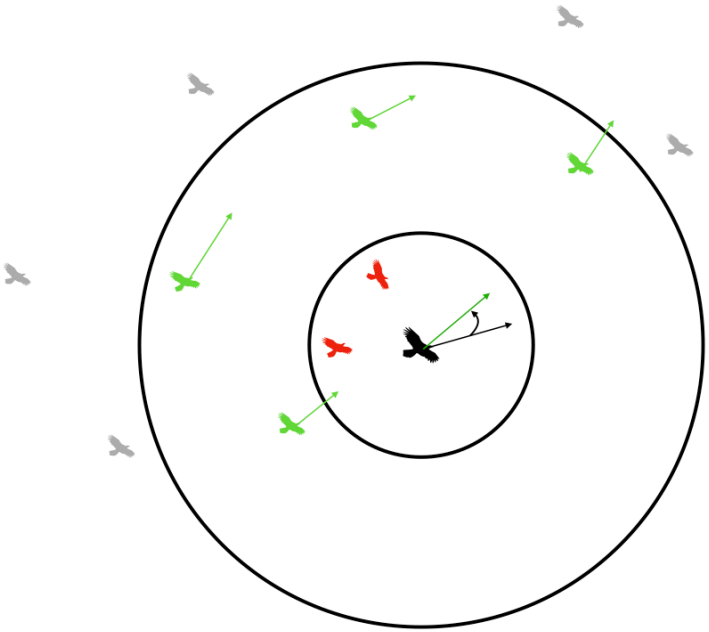
For boid b_k in N_{b_i}

$\mathbf{v} = \mathbf{v} + \mathbf{d}_k$

$\mathbf{v} = \mathbf{v}/|N_{b_i}|$ if $|N_{b_i}| > \mathbf{0}$

$\mathbf{v} = \text{align\_factor} \cdot \mathbf{v}$

Return $\mathbf{v}$



https://vanhunteradams.com/Pico/Animal_Movement/Boids-algorithm.html

Rule3: Cohesion

- Each boid steers gently toward the center of mass of other boids within its visible range

Procedure $\mathrm{Rule}_{\mathrm{coh}}(b_i, N_{b_i})$:

$\mathbf{p} = \mathbf{0}$

For boid b_k in N_{b_i}

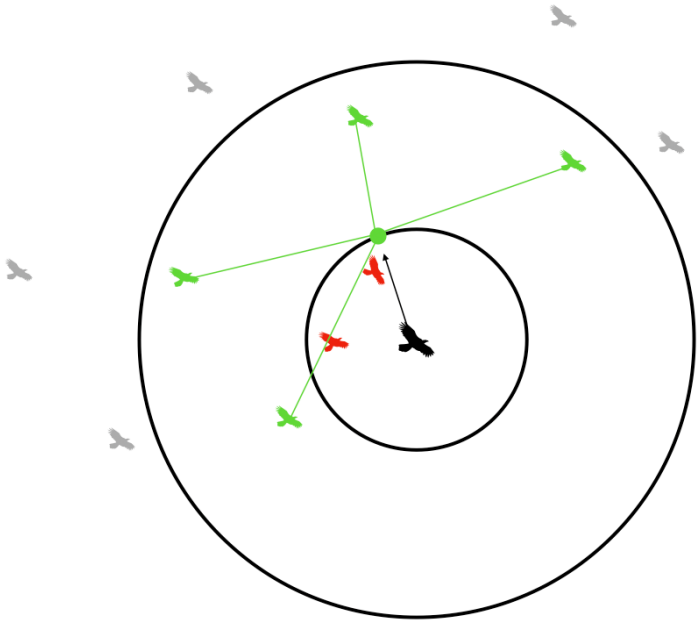
$\mathbf{p} = \mathbf{p} + \mathbf{p}_{\mathrm{k}}$

$\mathbf{p} = \mathbf{p}/\left|N_{b_i}\right|$ if $\left|N_{b_i}\right| > \mathbf{0}$

$\mathbf{v} = (\mathbf{p} - \mathbf{p}_{\mathrm{i}})$

$\mathbf{v} = \mathrm{cohesion\_factor} \cdot \mathbf{v}$

Return $\mathbf{v}$



https://vanhunteradams.com/Pico/Animal_Movement/Boids-algorithm.html

Handling Boundary

- To prevent simulated boids from being out of screen, hard/soft boundary rule can be applied

For each boid

$$p_x = \min(\max(p_x, x_{\min}), x_{\max})$$

$$p_y = \min(\max(p_y, y_{\min}), y_{\max})$$

Hard Boundary

For each boid

If $p_x < x_{\min}$ **then** $v_x = v_x + push\_factor$

If $p_x > x_{\max}$ **then** $v_x = v_x - push\_factor$

If $p_y < y_{\min}$ **then** $v_y = v_y + push\_factor$

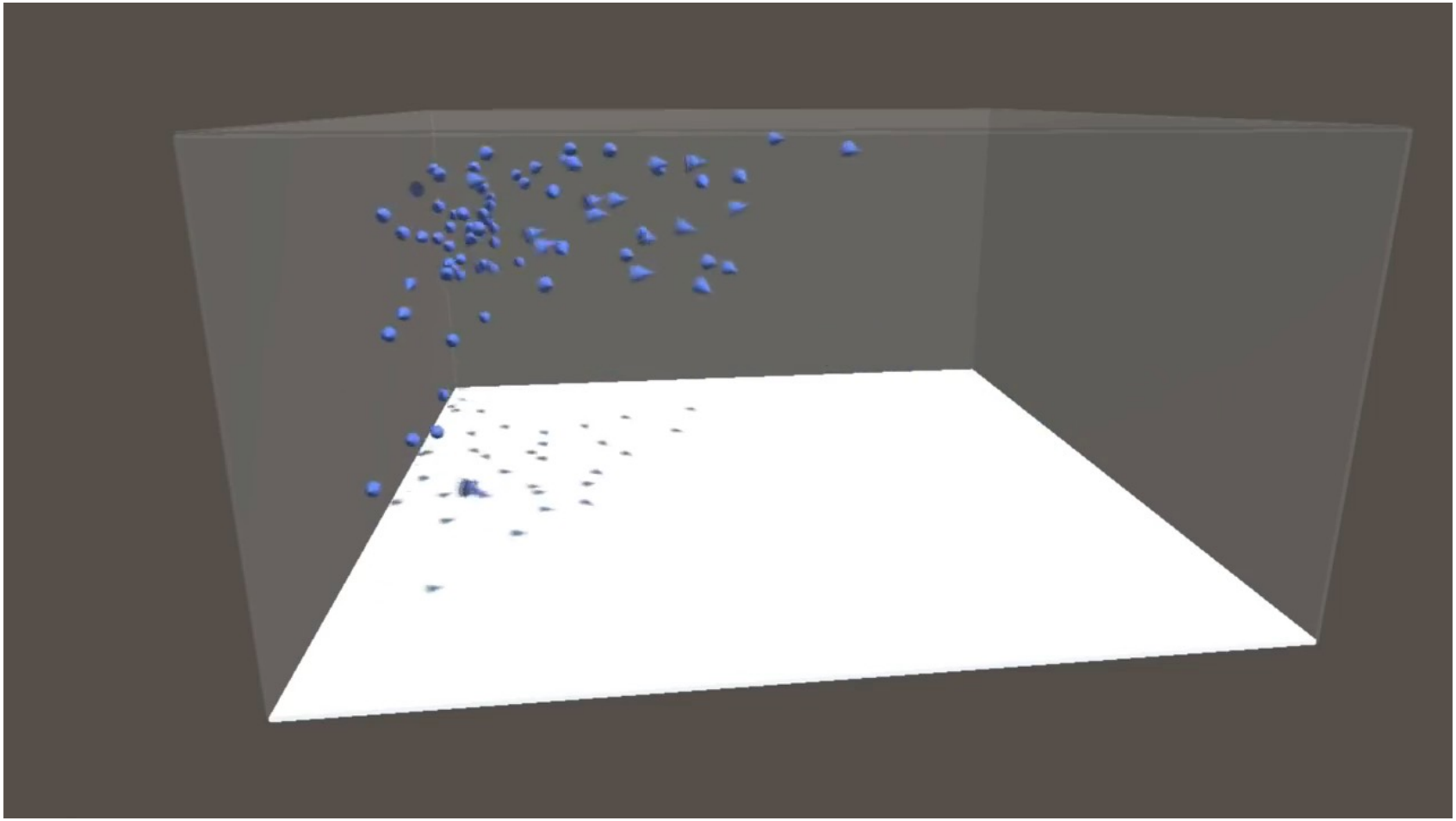
If $p_y > y_{\max}$ **then** $v_y = v_y - push\_factor$

Soft Boundary

Other Behaviors

- Predators
  - Predators are agents who want to break up the flock
- Non-favorite Locations
  - There could exist some dangerous static places which agents do not want to be close
- Perching
  - The boids often land and stay on the ground for a brief period of time before returning to the flock

A Good Example of Implementation of Boids Model



https://www.youtube.com/watch?v=bqtqltqcQhw&list=RDQMsCrzCVIzM9U&start_radio=1

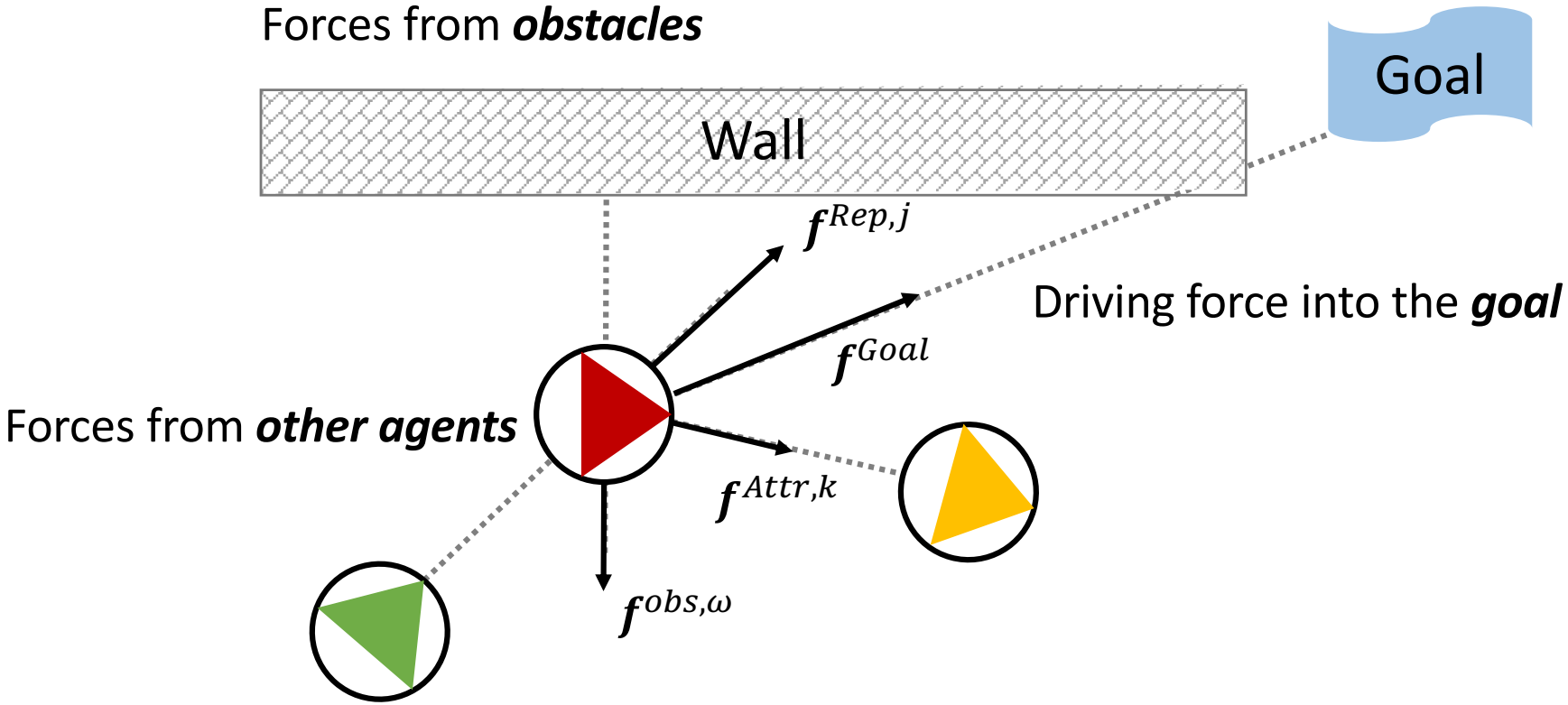
Social Force Model (SFM)

- Social forces are the ***forces*** are not directly exerted by the pedestrians' personal environment, but they are a measure for the internal motivations of the individuals to perform certain actions

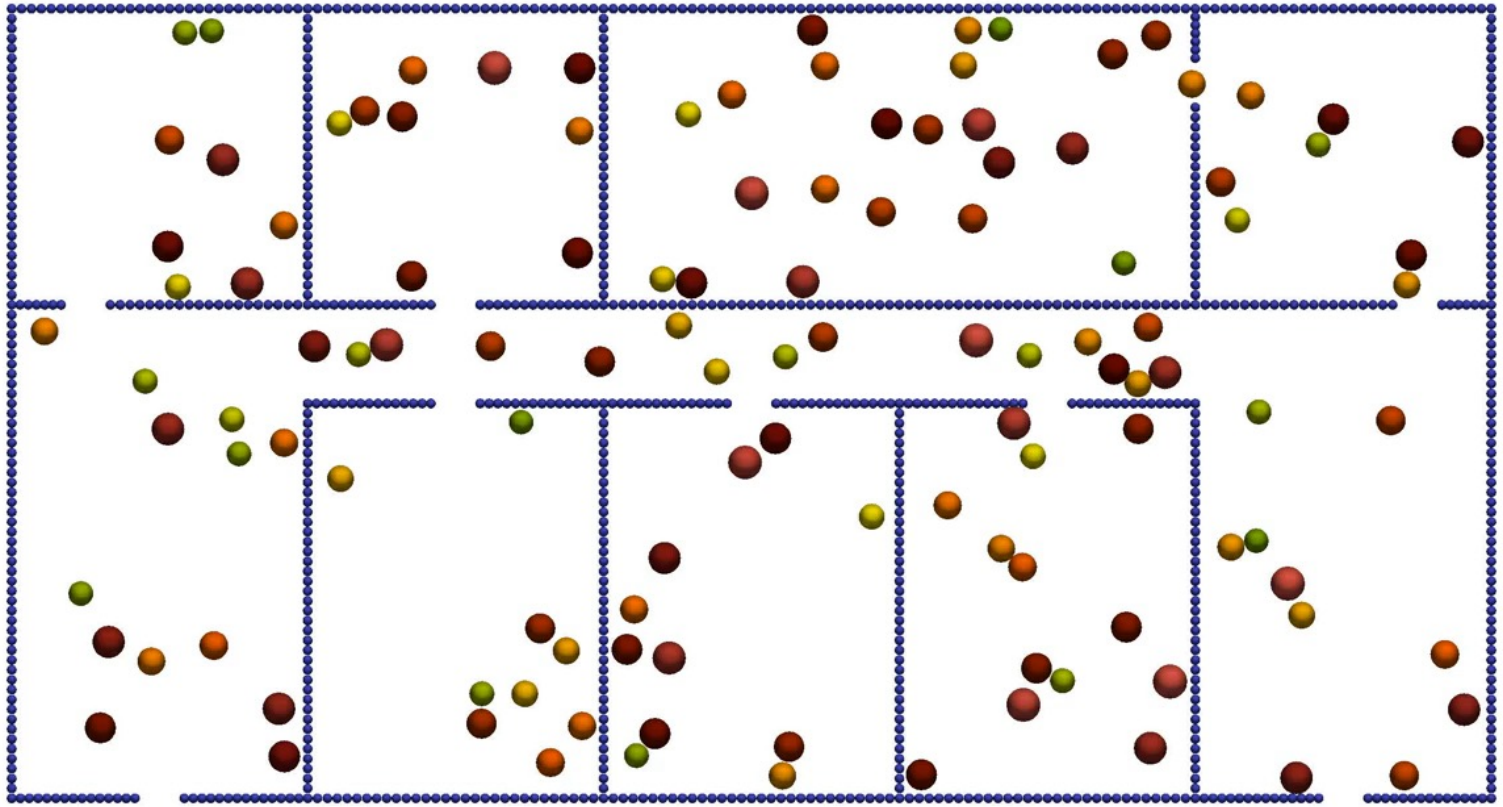
- All computations are performed in the level of ***acceleration*** because we are computing ***forces***

- First introduced by [Helbing et al. 1998] to model pedestrian dynamics

Social Force Model (SFM)



Social Force Model (SFM)



https://www.youtube.com/watch?v=lg3CzDlIbt4

Social Force Model (SFM)

Initialize states
Set desired goals
Until max-step reached
    For every agents

$$m\boldsymbol{a}(t) = \boldsymbol{f}^{Goal} + \sum_j \boldsymbol{f}^{Rep,j} + \sum_k \boldsymbol{f}^{Attr,k} + \sum_\omega \boldsymbol{f}^{obs,\omega}$$

$$\mathbf{v}_i(t) \leftarrow \mathbf{v}_i(t) + \boldsymbol{a}_i(t)\,\Delta t$$

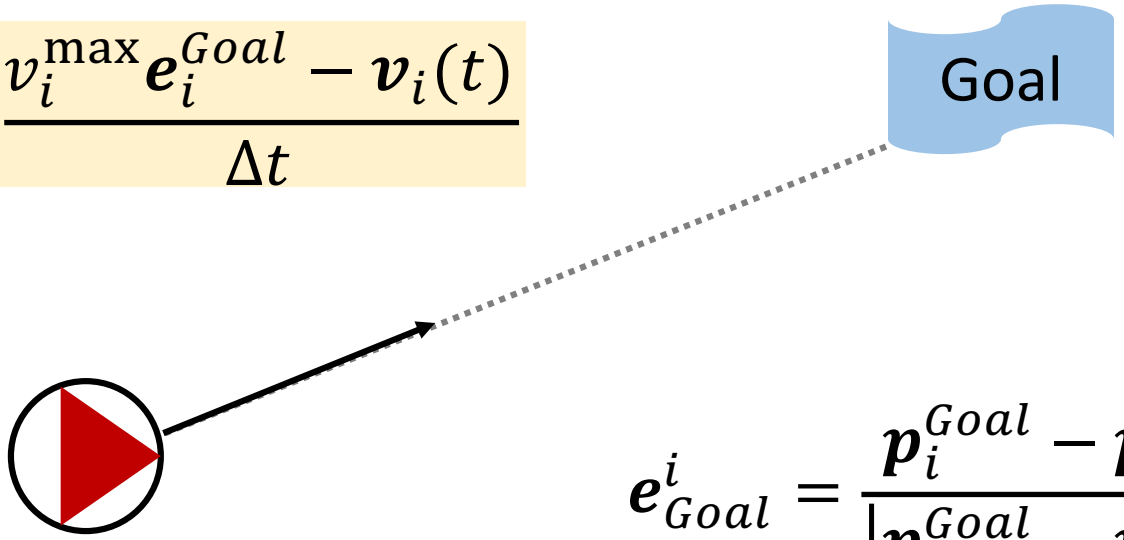
$$\mathbf{p}_i(t) \leftarrow \mathbf{p}_i(t) + \mathbf{v}_i(t)\Delta t$$

- $\boldsymbol{f}^{Goal}$: the motivation of pedestrian to reach to the desired goal position
- $\boldsymbol{f}^{Rep,j}$: the repulsive force between an agent and j-th agent
- $\boldsymbol{f}^{Attr,k}$: the attractive force into other agents (e.g., friends)
- $\boldsymbol{f}^{obs,\omega}$: the interaction between an agent and k-th obstacle (e.g., walls)

Force to the Desired Position

- An agent wants to reach a certain destination as soon as possible

$$\boldsymbol{f}_i^{Goal} = \frac{v_i^{\max}\boldsymbol{e}_i^{Goal} - \boldsymbol{v}_i(t)}{\Delta t}$$

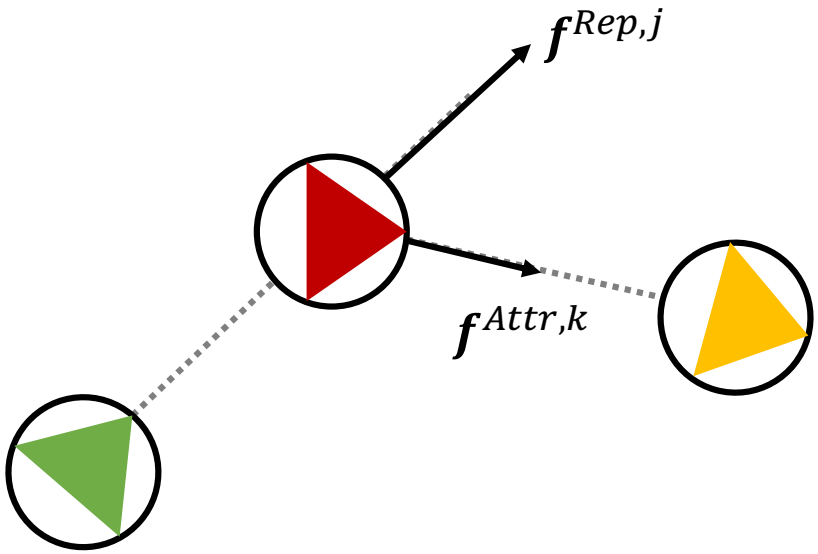


$$\boldsymbol{e}_{Goal}^{i} = \frac{\boldsymbol{p}_i^{Goal} - \boldsymbol{p}_i(t)}{\left|\boldsymbol{p}_i^{Goal} - \boldsymbol{p}_i(t)\right|}$$

$v_i^{\max}$: a tunable parameter

Forces from Other Agents

- An agent normally feels increasingly uncomfortable the closer he/she gets to a strange person, who may react in an aggressive way
- Agents are sometimes attracted by other persons (friends, street artists, etc.) or objects (e.g. window displays)

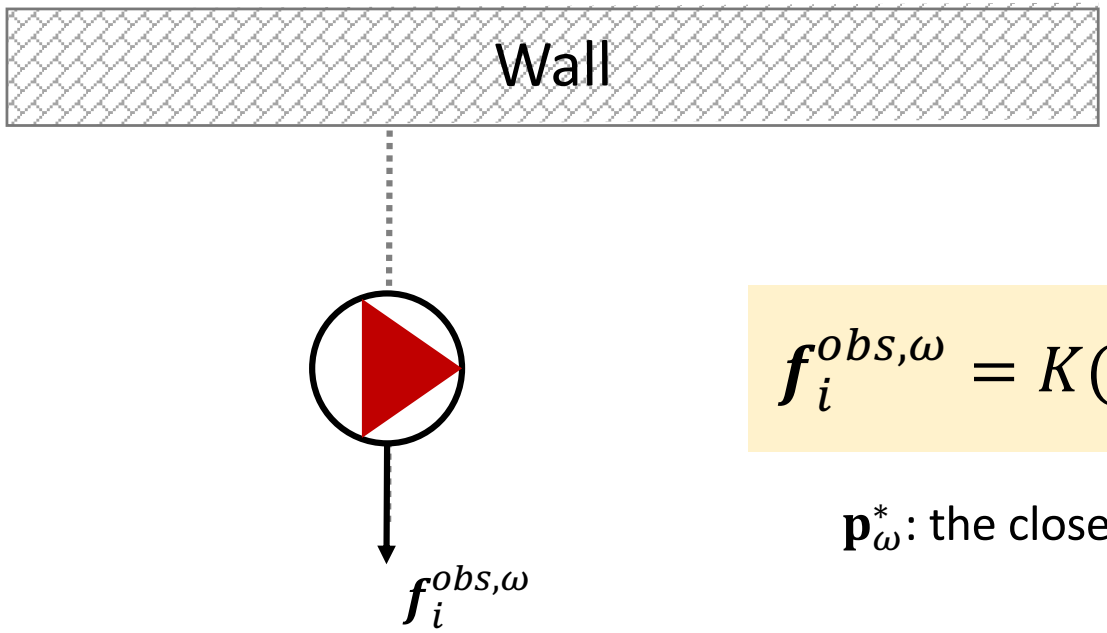


$$\boldsymbol{f}_i^{Rep,j} = K(|\mathbf{p}_i - \mathbf{p}_j|)\,\frac{\mathbf{p}_i - \mathbf{p}_j}{|\mathbf{p}_i - \mathbf{p}_j|}$$

$$\boldsymbol{f}_i^{Attr,k} = -K(|\mathbf{p}_i - \mathbf{p}_k|)\,\frac{\mathbf{p}_i - \mathbf{p}_k}{|\mathbf{p}_i - \mathbf{p}_k|}$$

Forces from Obstacles

- An agent tries to keep away from obstacles to avoid collision because in reality a person will get hurt if he/she collides with obstacles (e.g. walls)



$$f_i^{obs,\omega} = K(|\mathbf{p}_i - \mathbf{p}_\omega^*|)\frac{\mathbf{p}_i - \mathbf{p}_\omega^*}{|\mathbf{p}_i - \mathbf{p}_\omega^*|}$$

$\mathbf{p}_\omega^*$: the closest point on the obstacle

Other Social Forces

- Personal Space
- Relative velocities between individuals
- Evasive force

Velocity-based Model

- Modeling potential avoidance/collision maneuvers in the ***velocity space***

- It is often more stable than force-based model, in other words, larger-time step could be used

- Assuming linear motion between time steps, so it could occur error if object motions are non-linear or the time step is too extreme

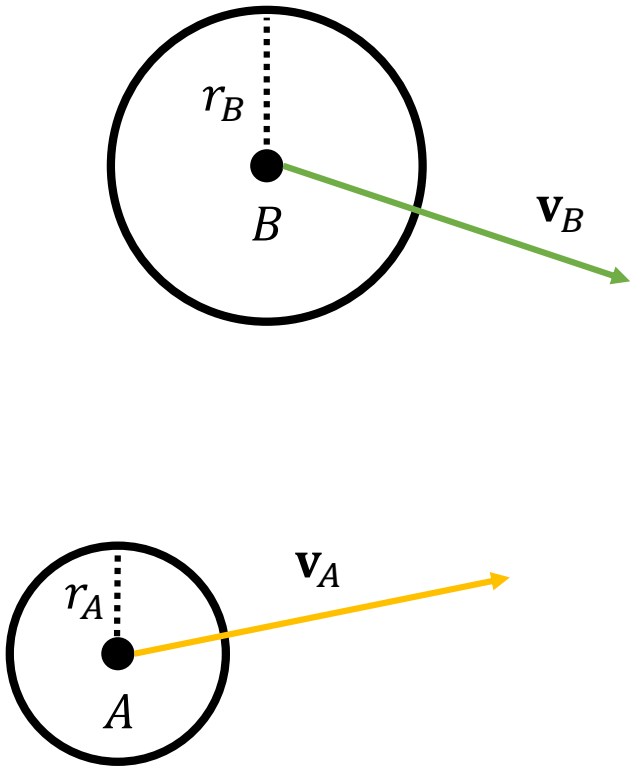
Velocity Obstacle (VO)

- Predicting potential collision by using ***relative velocities*** among agents

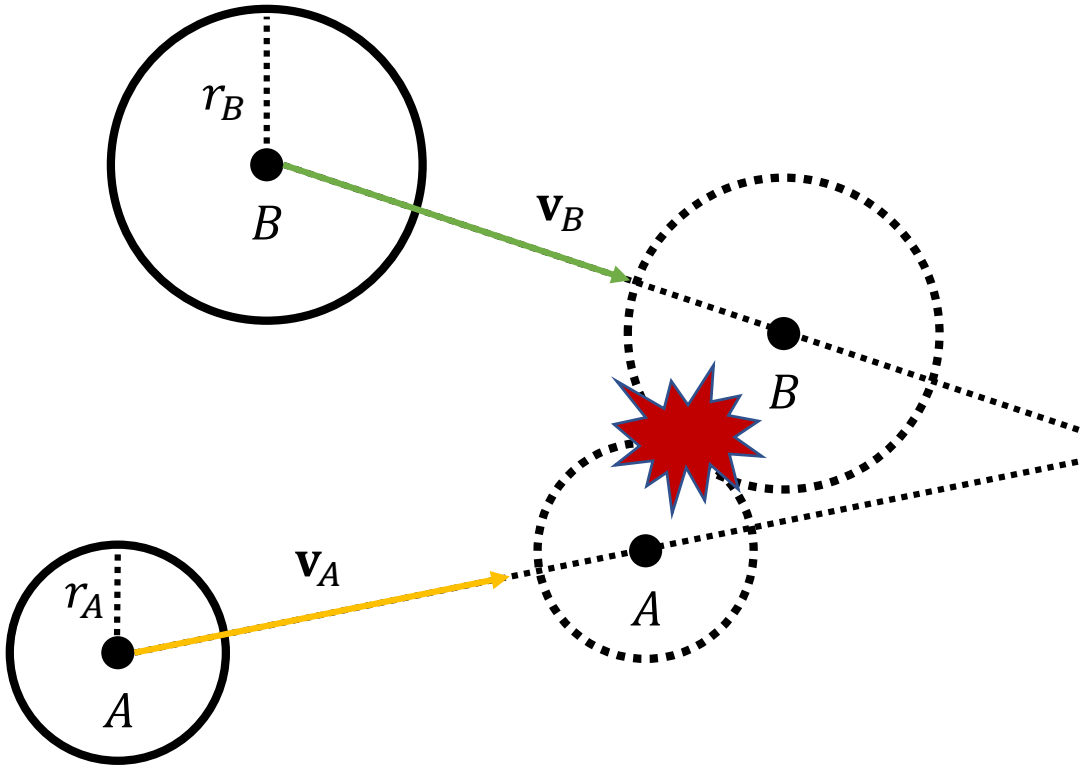
- Assuming that the velocity remains constant for each agent, (i) a velocity set ***velocity obstacles (VO)*** occurring collision at some point in the future and (ii) another velocity set ***reachable velocity (RV)*** which is reachable by agent's capability can be computed. Then, we take a velocity v where $v \in RV$ and $v \notin VO$

Fiorini and Shiller, Motion Planning in Dynamic Environments using Velocity Obstacles, 1998

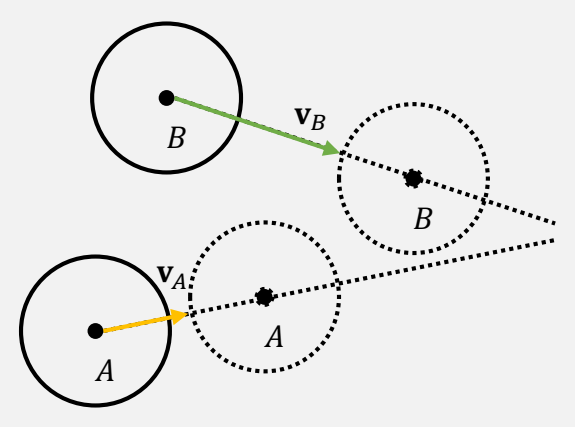
Velocity Obstacle (VO)



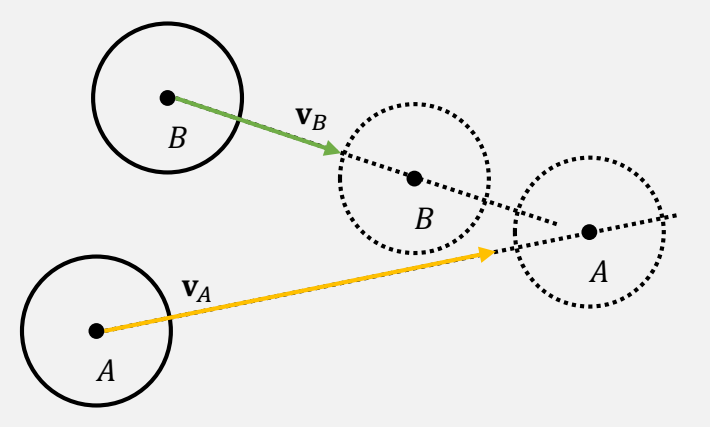
Velocity Obstacle (VO)



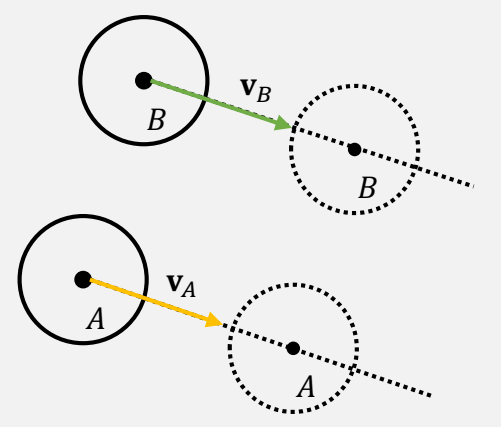
Ways to Avoid Collision



(a) Slow down



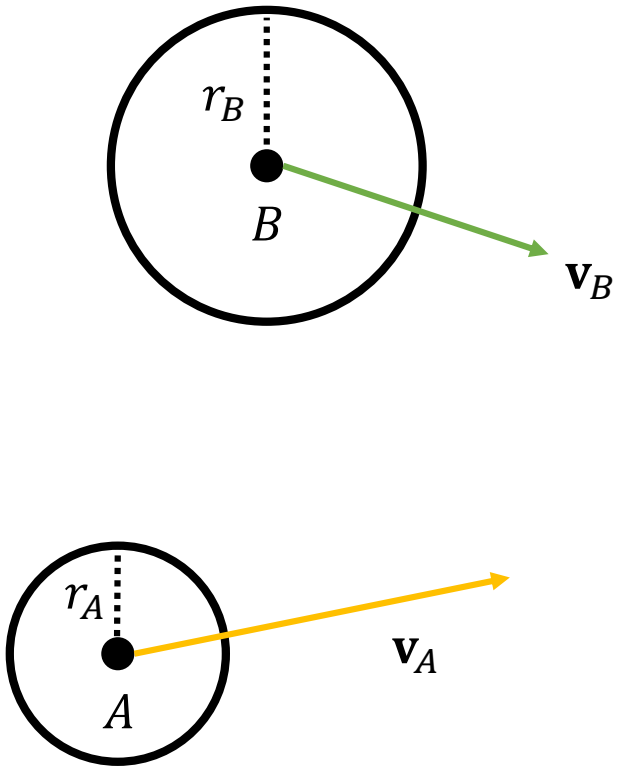
(b) Speed up



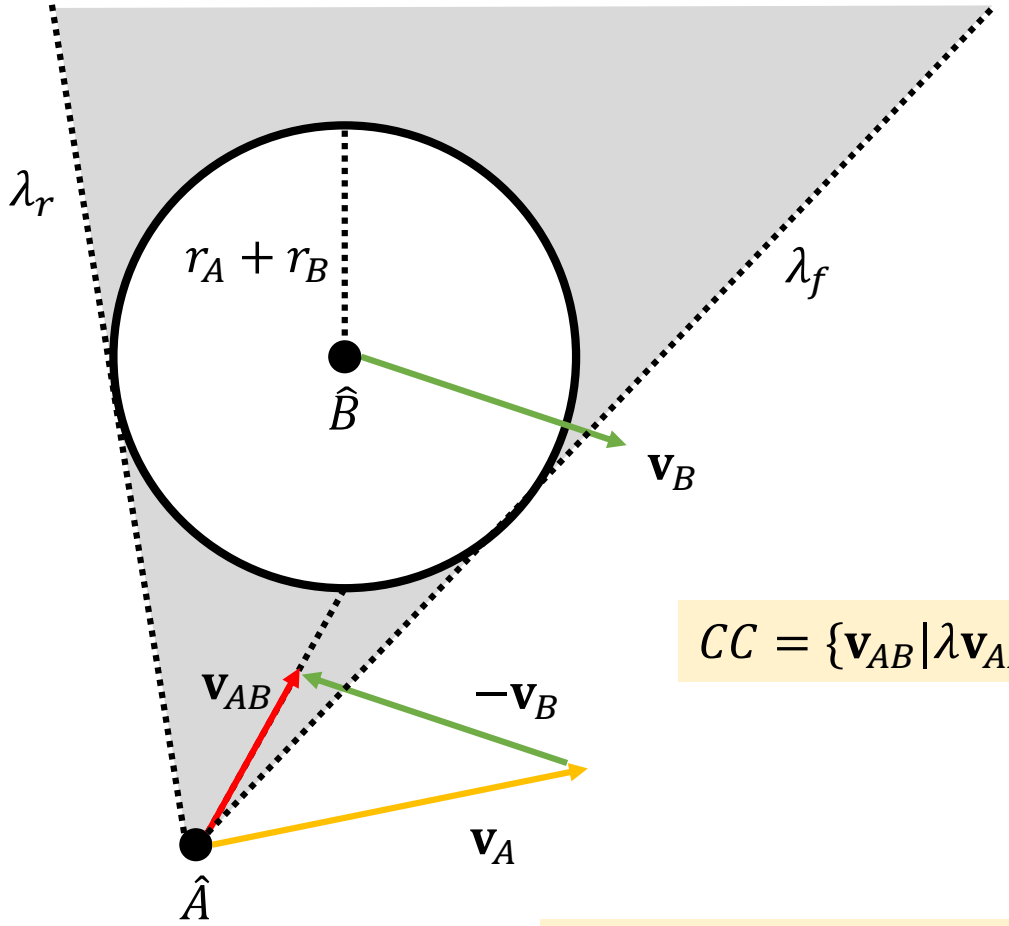
(c) Change direction

+ Combinations of (a), (b), and (c)

Velocity Obstacle (VO)



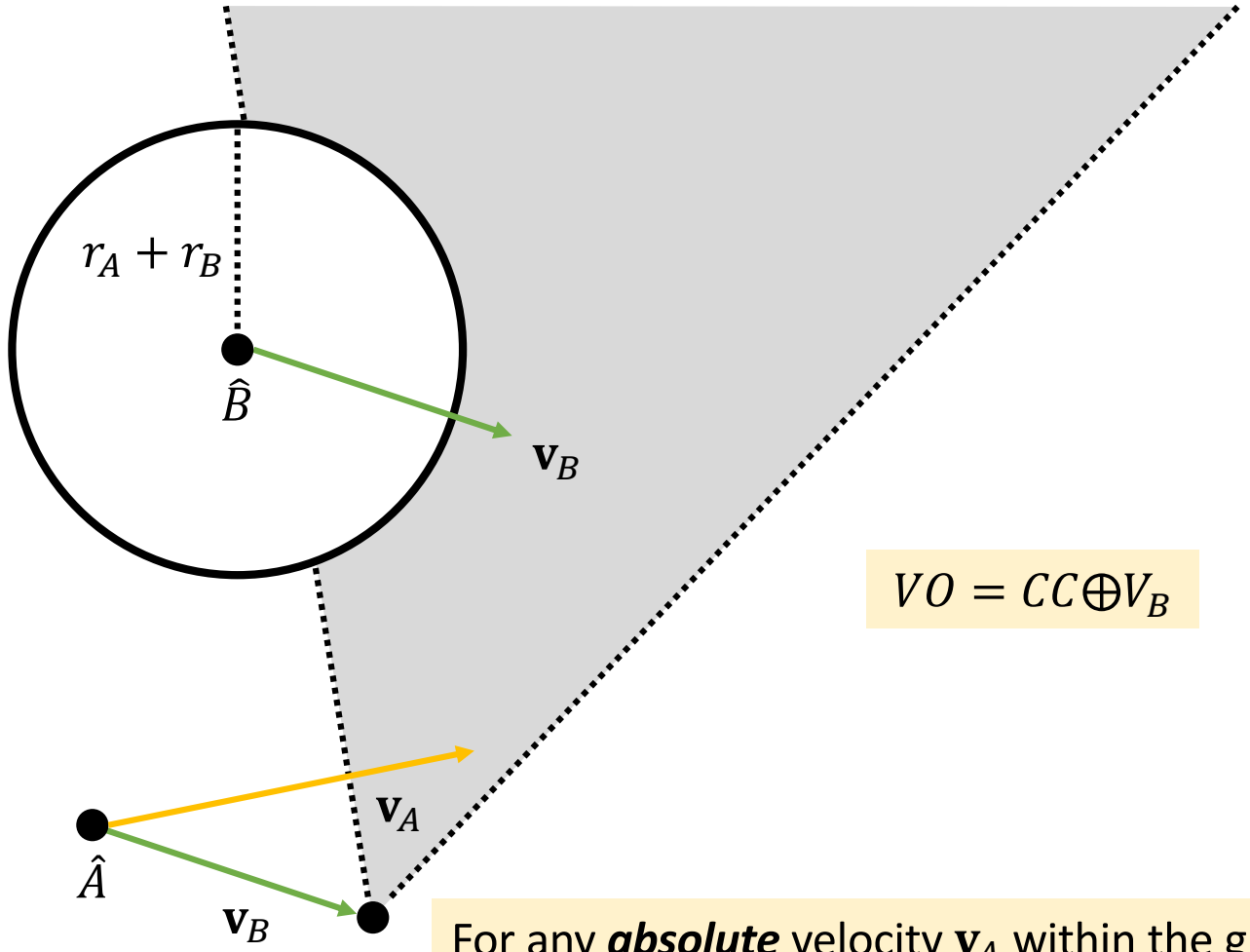
Collision Cone (CC)



$CC = \{\mathbf{v}_{AB} | \lambda \mathbf{v}_{AB} \cap \hat{B} \neq \emptyset\}$ where $\lambda \in R$

For any ***relative*** velocity $\mathbf{v}_{AB}$ within the gray zone (***Collision Cone***), the agent A and the agent B will eventually collide at some point in the future

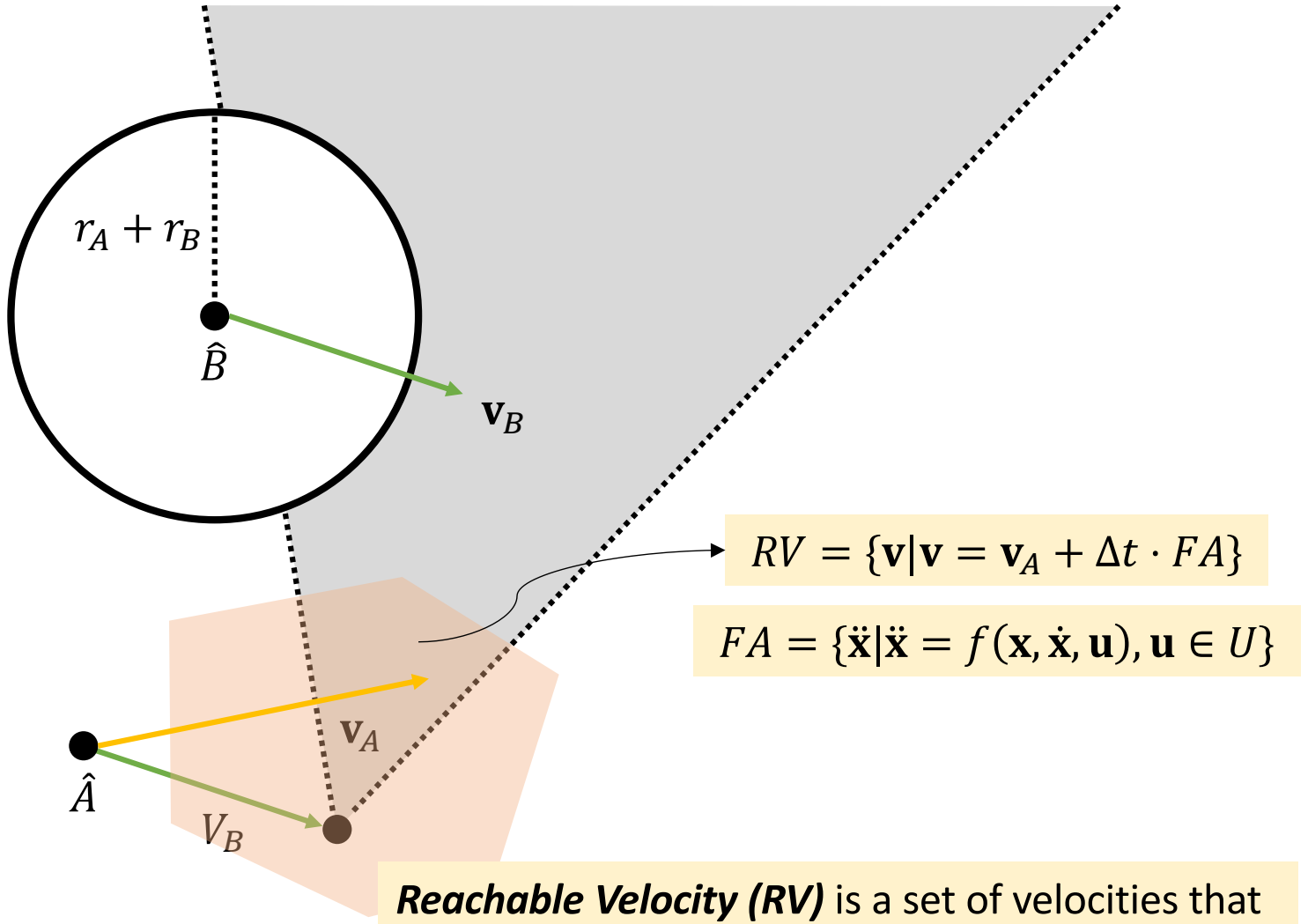
Velocity Obstacle (VO)



$$VO = CC \oplus V_B$$

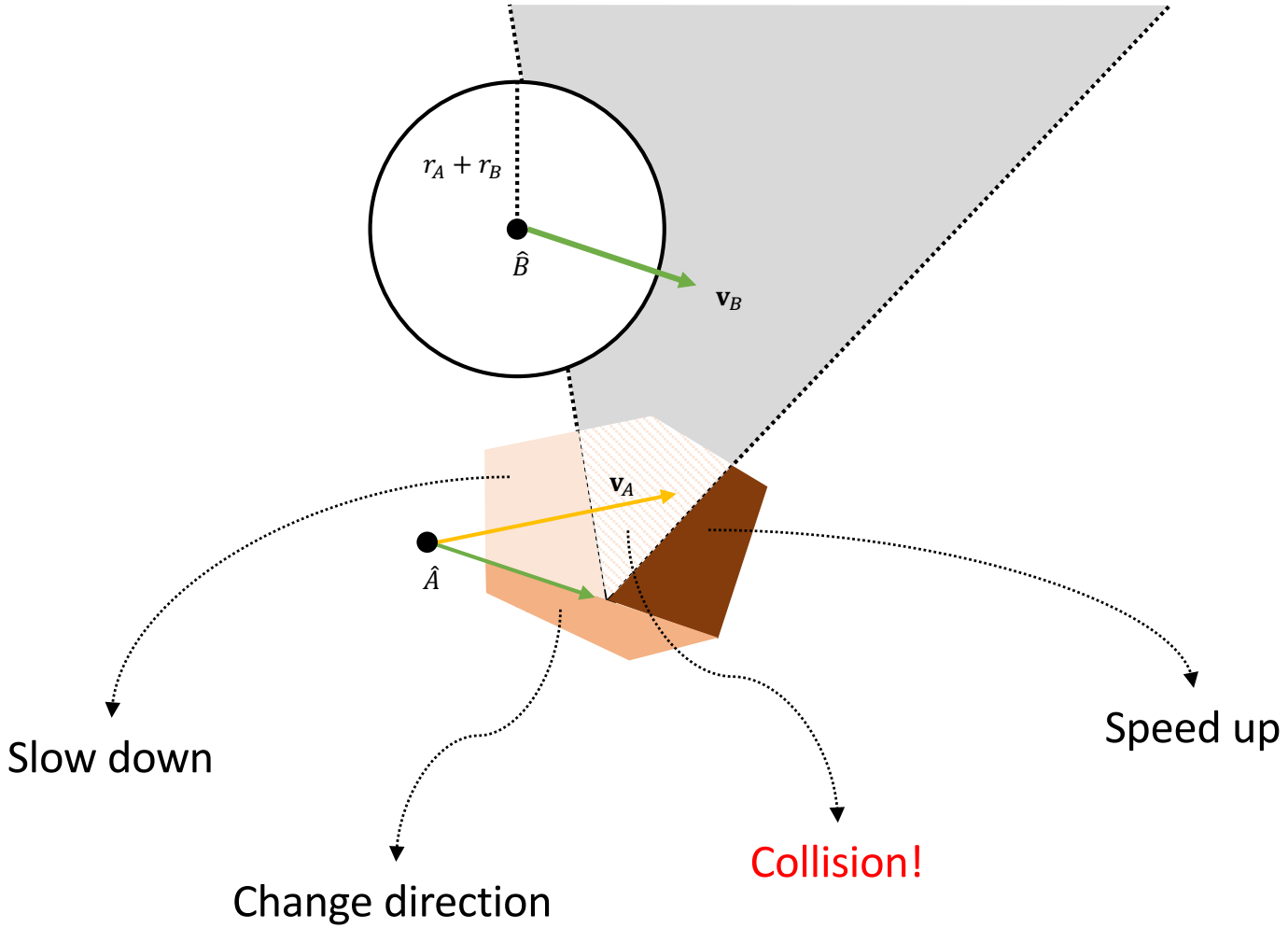
For any ***absolute*** velocity $\mathbf{v}_A$ within the gray zone (***Velocity Obstacle***), the agent A and the agent B will eventually collide at some point in the future

Reachable Velocity (RV)

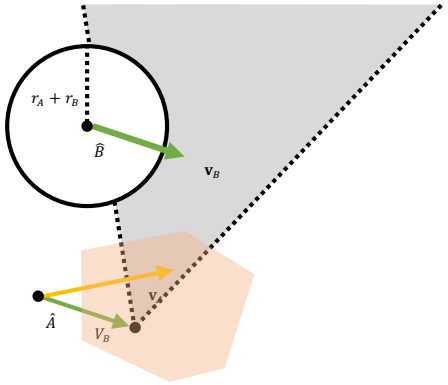


Reachable Velocity (RV) is a set of velocities that can be reachable by the agent's actuation ability

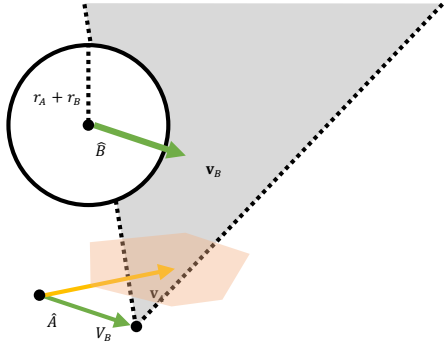
Reachable Velocity (RV)



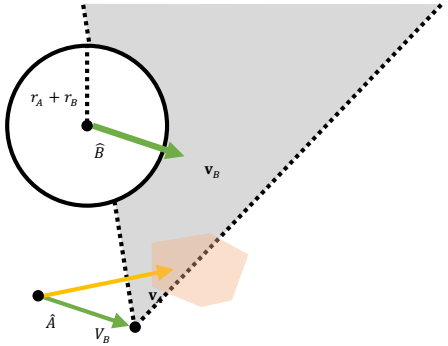
Reachable Velocity (RV)



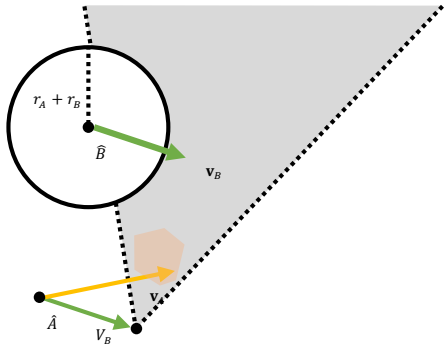
All options exist



No “change direction” option



Only “speed up” exists

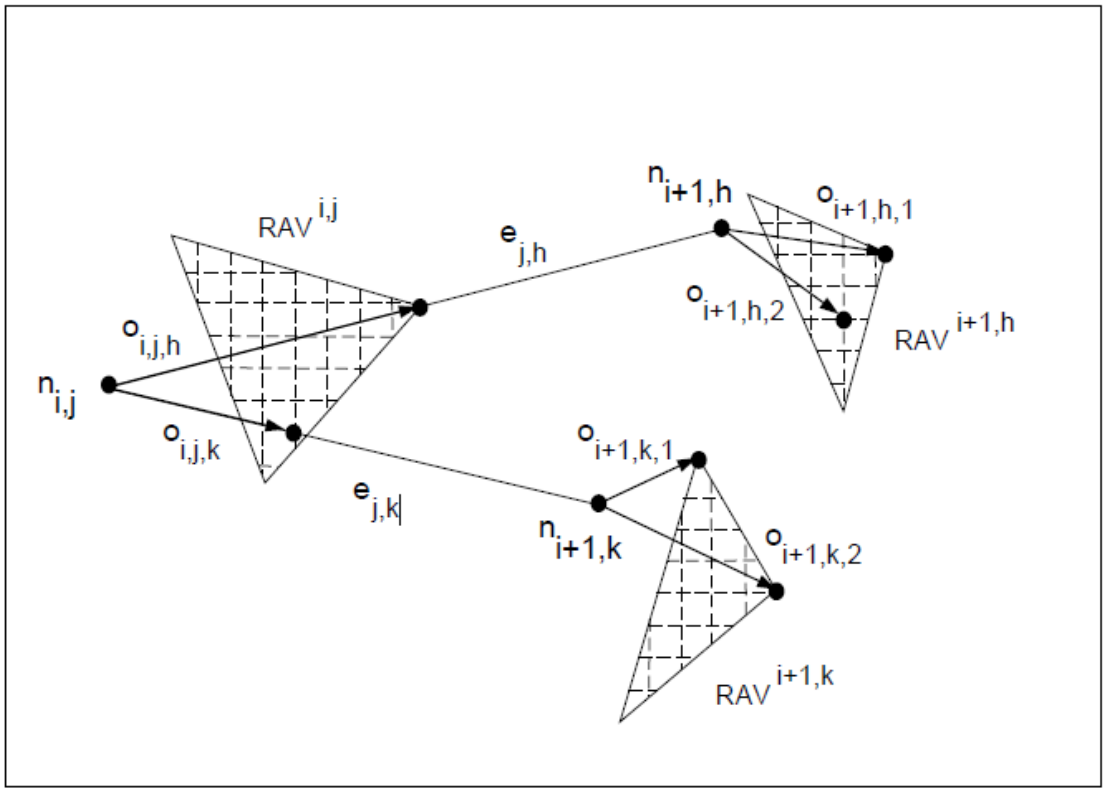


No option exists
(Collision is inevitable)

Computing the Avoidance Trajectories

- Global search
  - Discretize RV then check all collision-avoidable (RAV) options by expanding a game tree until the end of simulation time, then choose a collision-free trajectory
- Heuristic search
  - For on-line applications, the trajectory can be generated incrementally by expanding only the node corresponding to the current agent position, and generating only one branch per node, using some heuristics to choose among all possible branches

Global Search



Heuristic Search

- Choose the highest avoidance velocity along the line to the goal (TG strategy)
- Select the maximum avoidance velocity within some specified angle from the line to the goal so that the agent moves at high speed even if it does not aim directly at the goal (MV strategy)
- Select the velocity that avoids the obstacles according to their perceived risk (ST strategy - e.g. taking rear velocities in a “small car vs. big truck” scenario)
- Combination of some or all of the above strategies

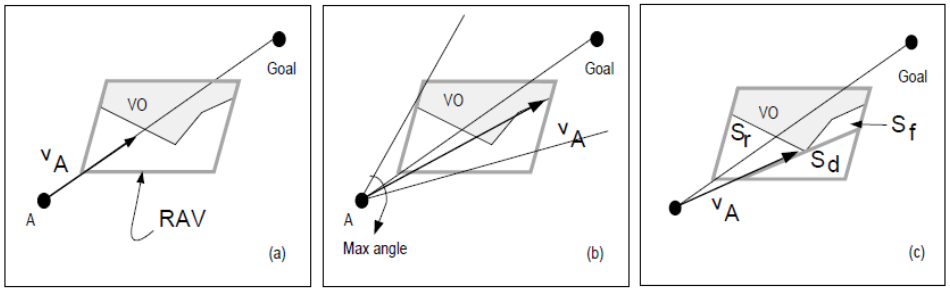
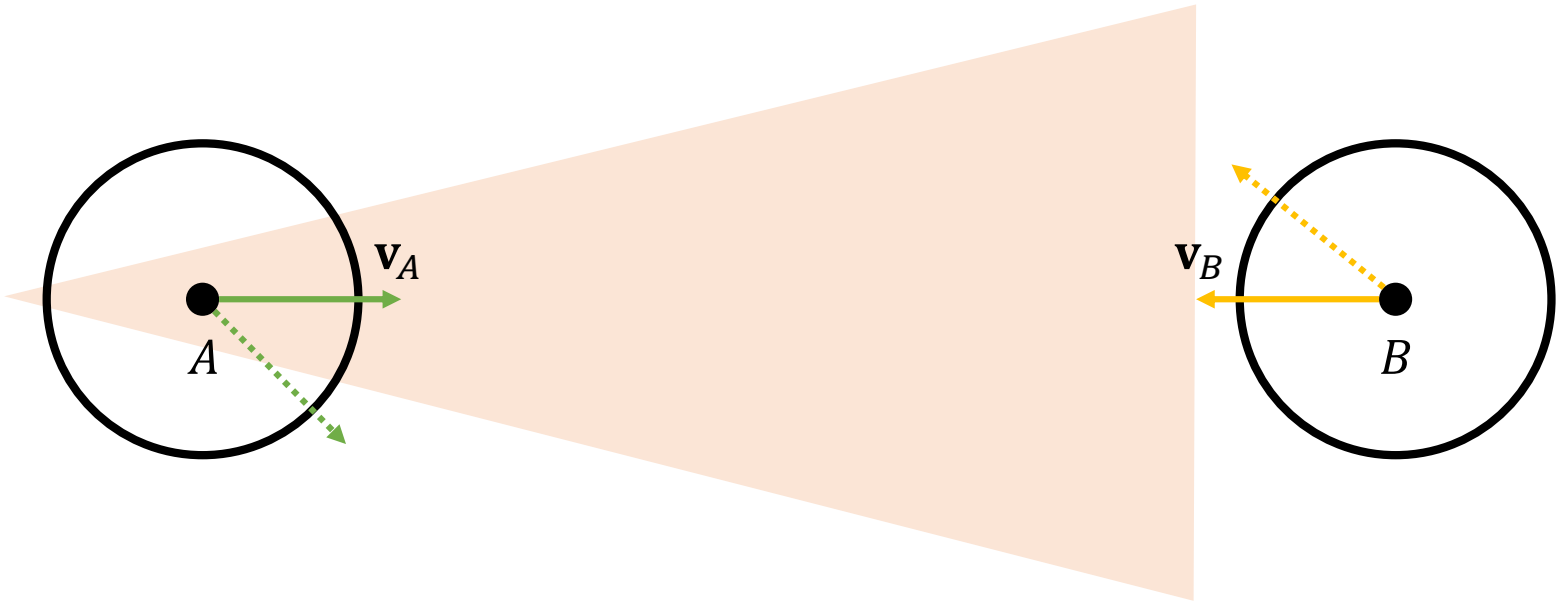
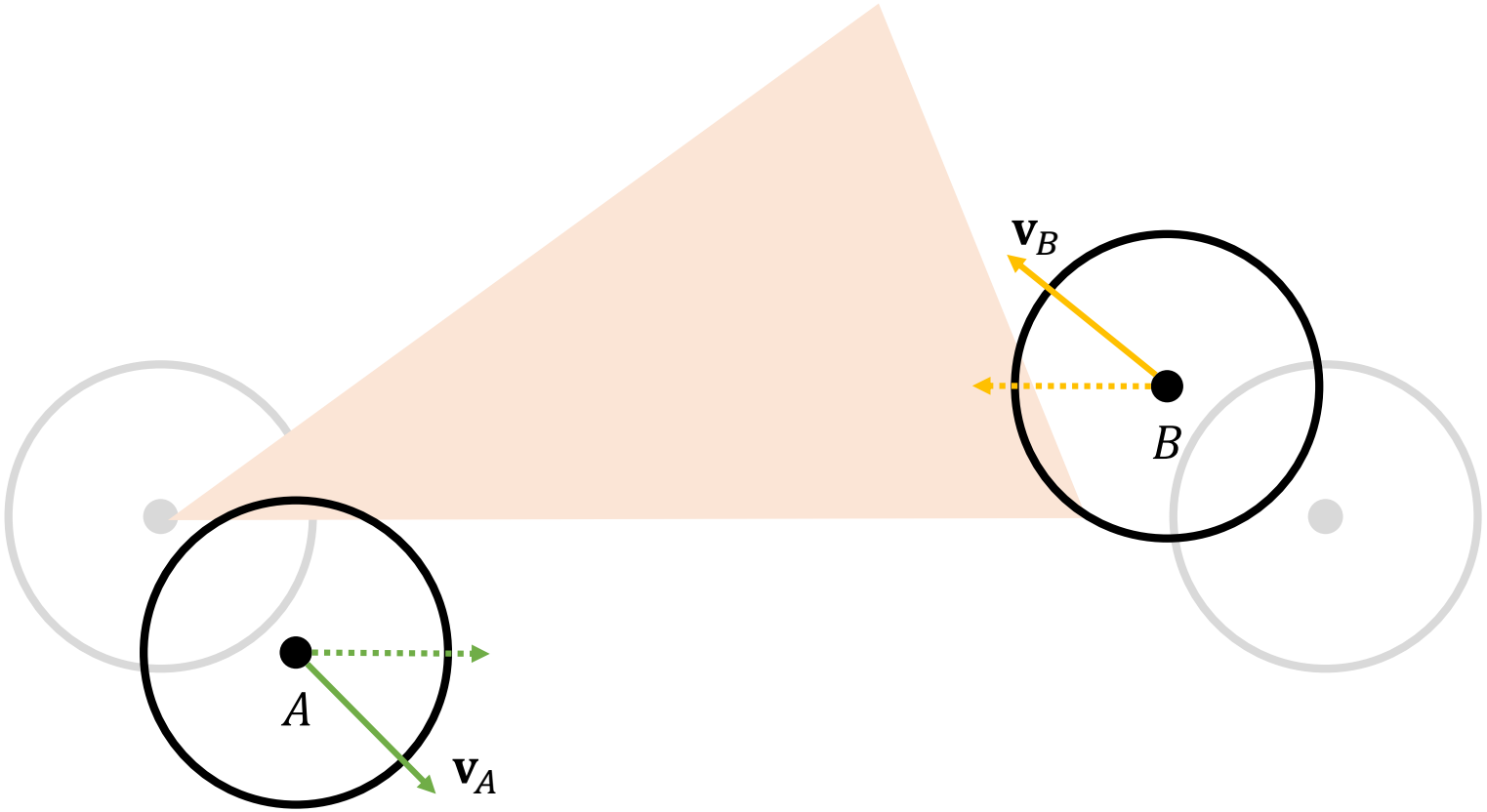


Figure 14: **a**: TG strategy. **b**: MV strategy. **c**: ST strategy.

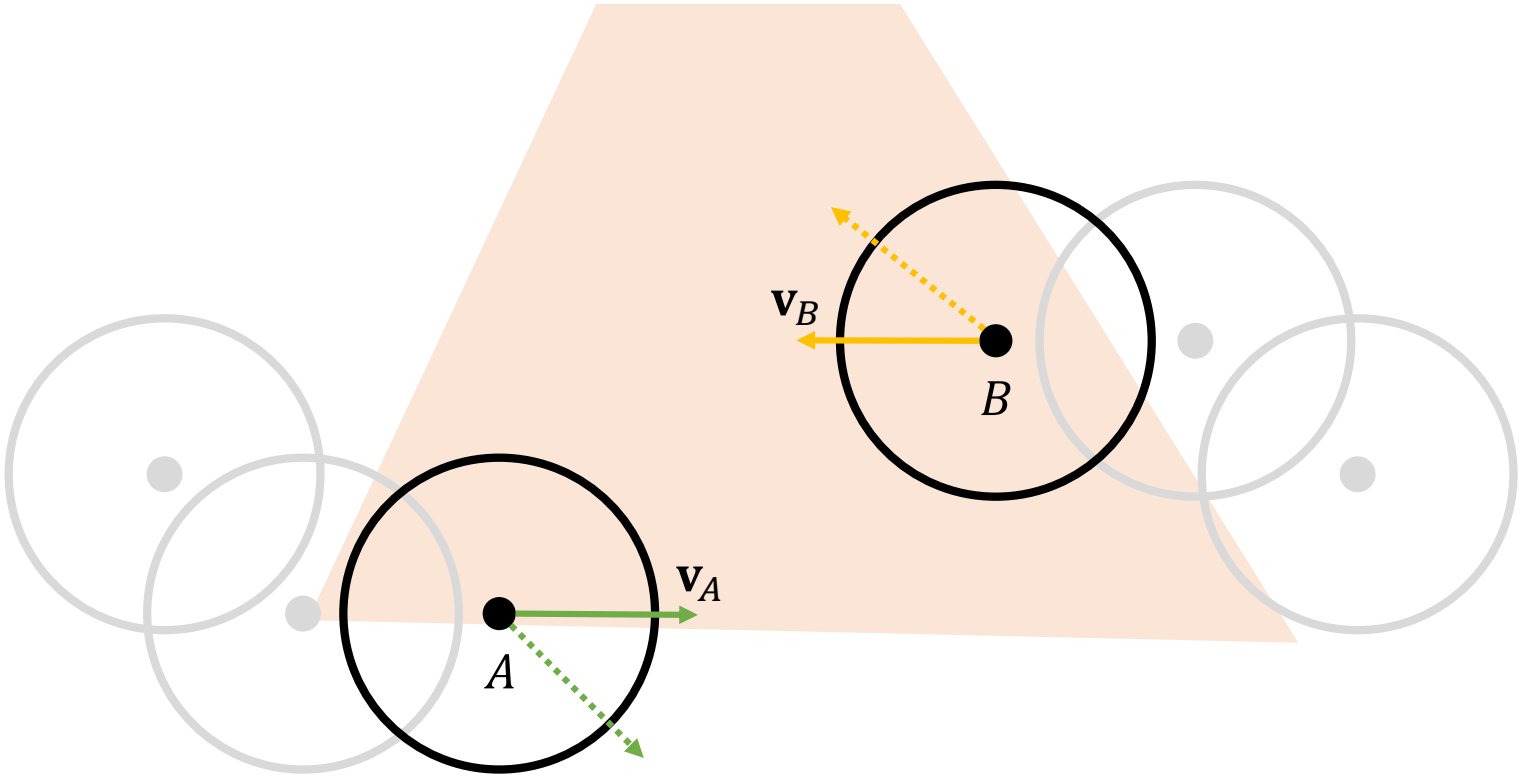
Oscillation: A Problem in VO



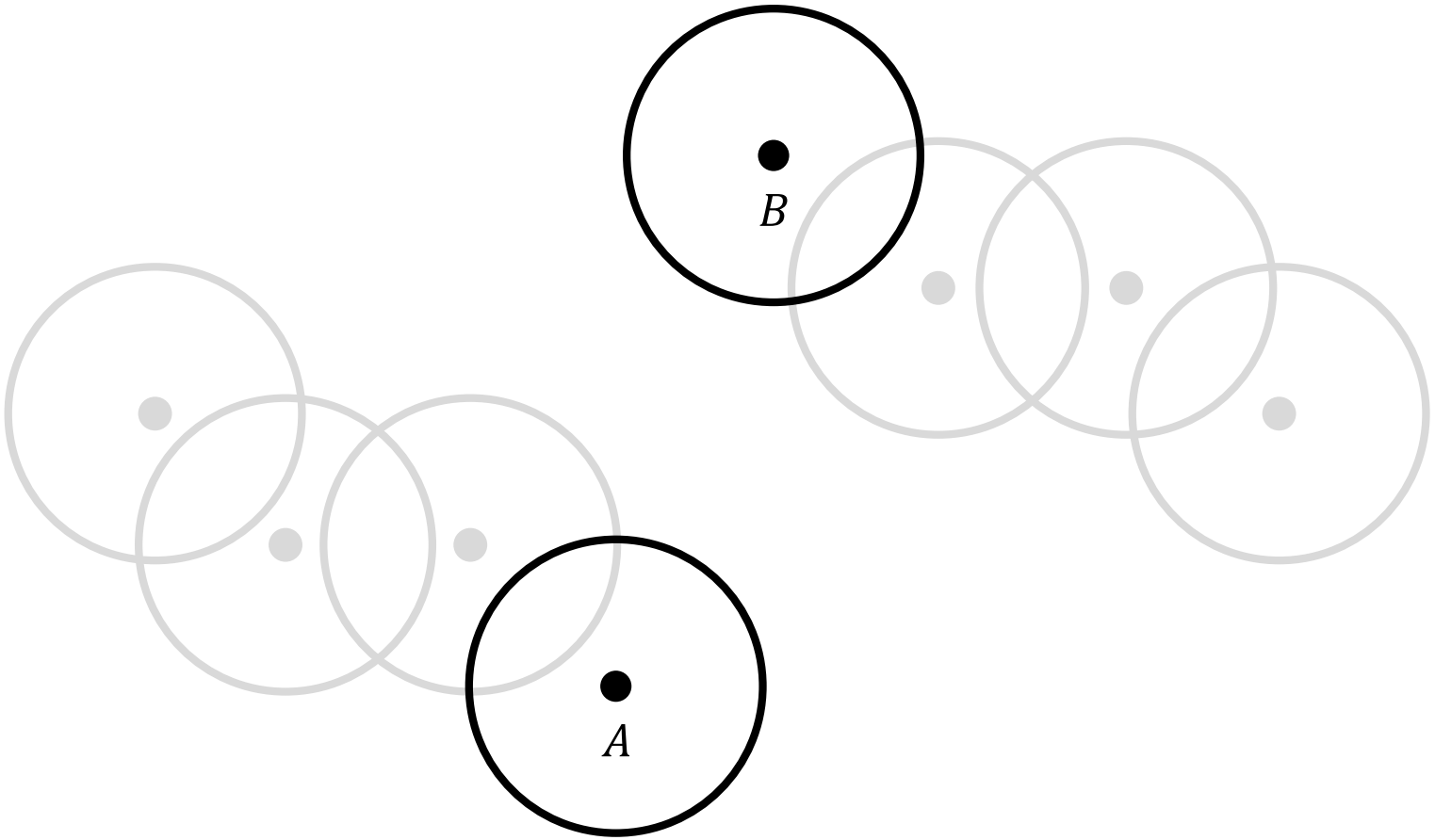
Oscillation: A Problem in VO



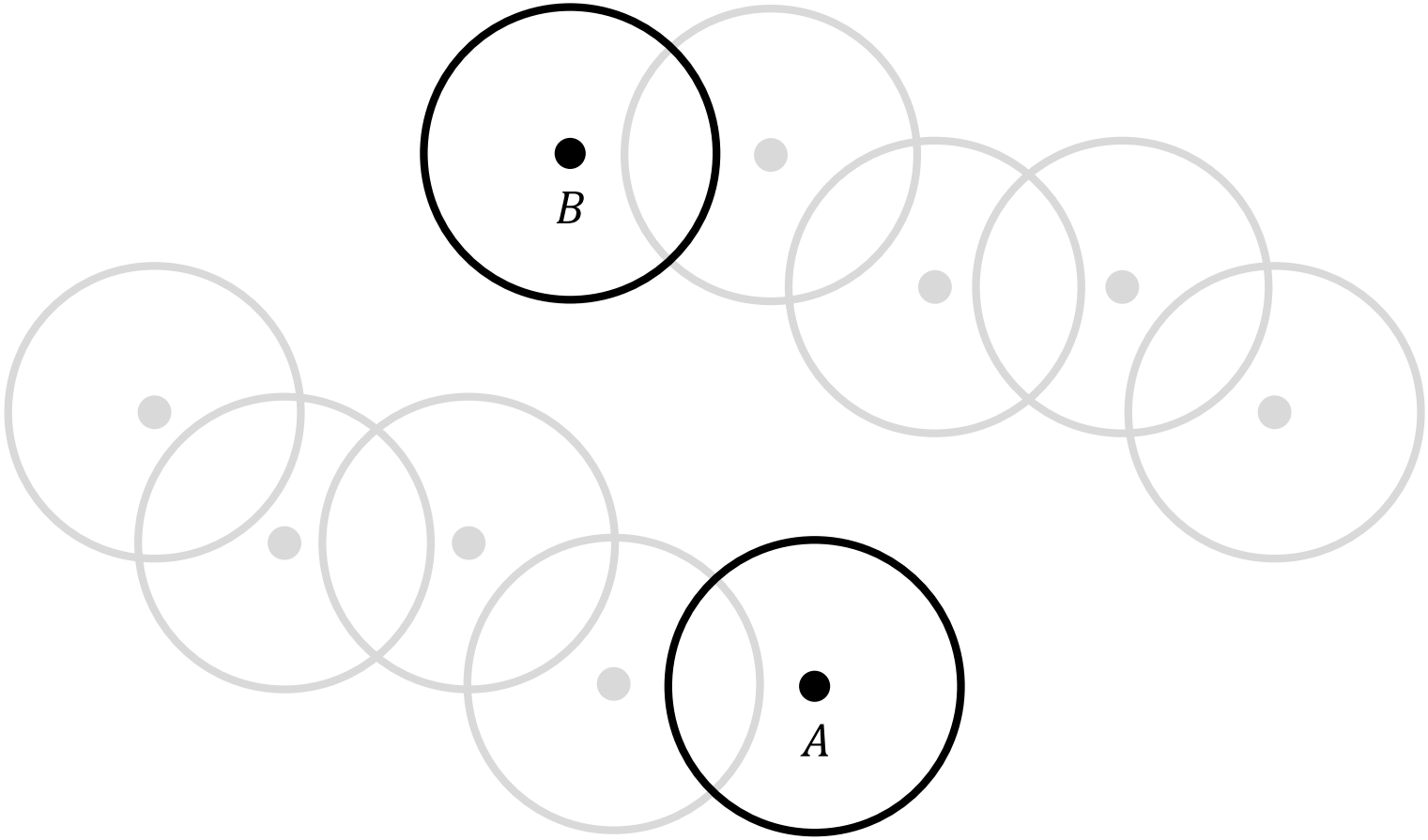
Oscillation: A Problem in VO



Oscillation: A Problem in VO



Oscillation: A Problem in VO



Reciprocal Velocity Obstacle (RVO)

- It was proposed by [Berg et al. 2008], which was published at ICRA, to mitigate the oscillation artifacts of VO

- It provides a simple approach to safely and smoothly navigate multiple agents amongst each other without explicit communication between them

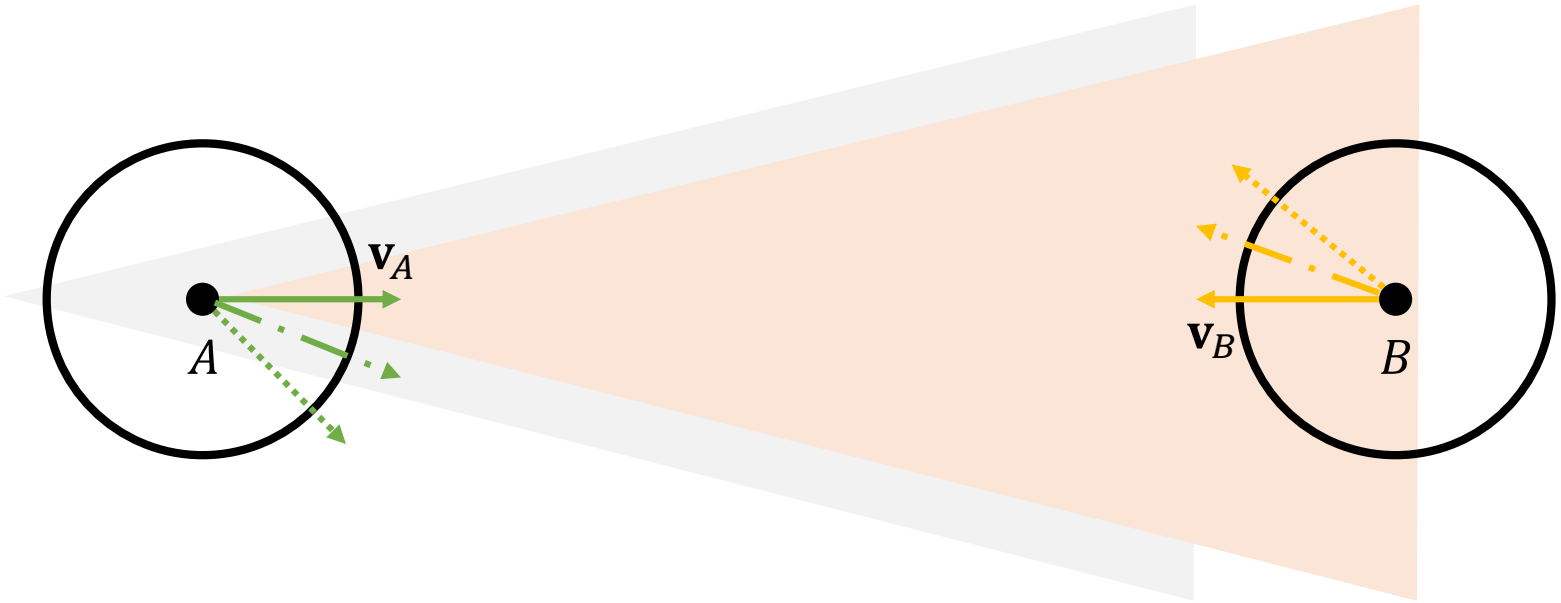
Reciprocal Velocity Obstacle (RVO)

- The basic idea is simple: instead of choosing a new velocity for each agent that is outside the other agent's velocity obstacle, it chooses a new velocity that is the ***average*** of (i) its current velocity and (ii) a velocity that lies outside the other agent's velocity obstacle

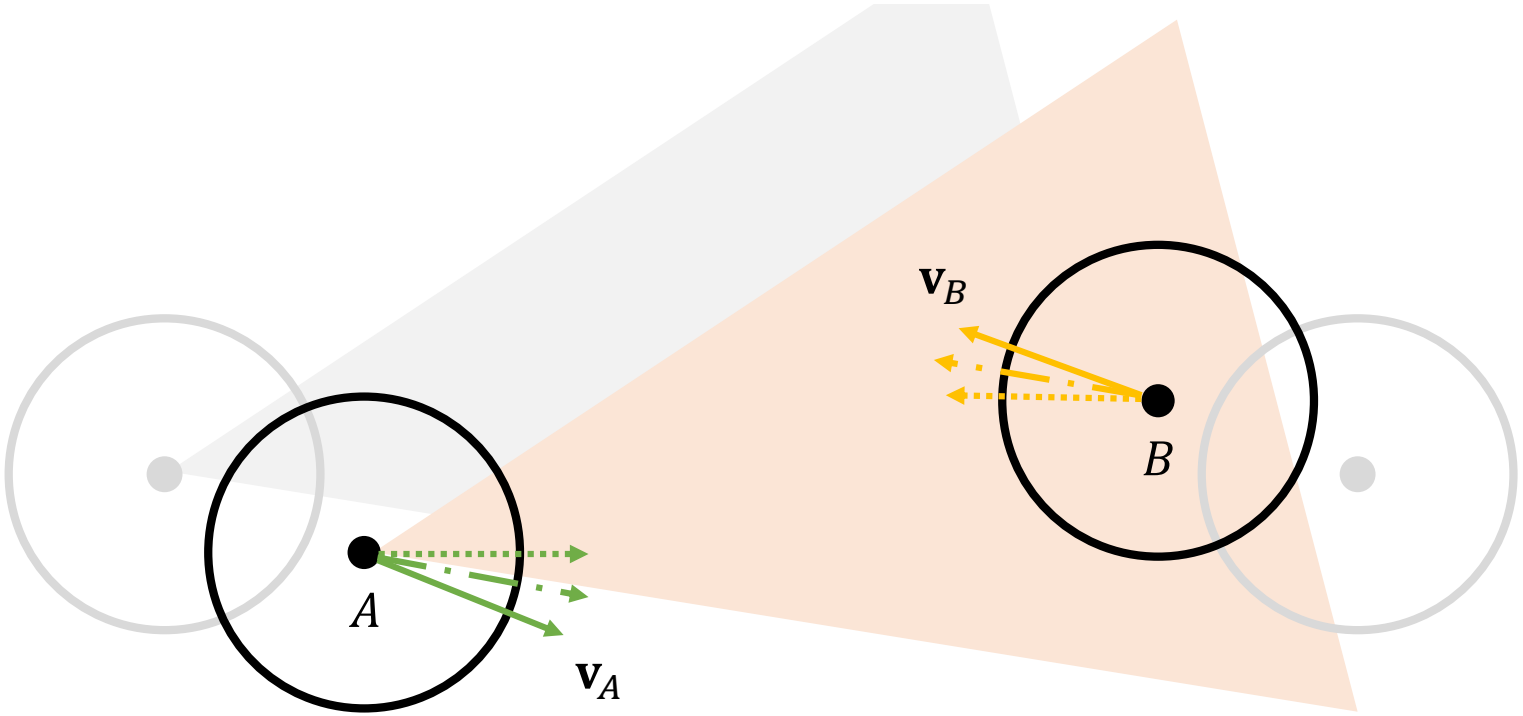
$$RVO = \{\mathbf{v}' | 2\mathbf{v}' - \mathbf{v} \in VO\}$$

- It can geometrically be interpreted as the velocity obstacle VO that is translated such that its apex lies at $\frac{\mathbf{v}_A + \mathbf{v}_B}{2}$

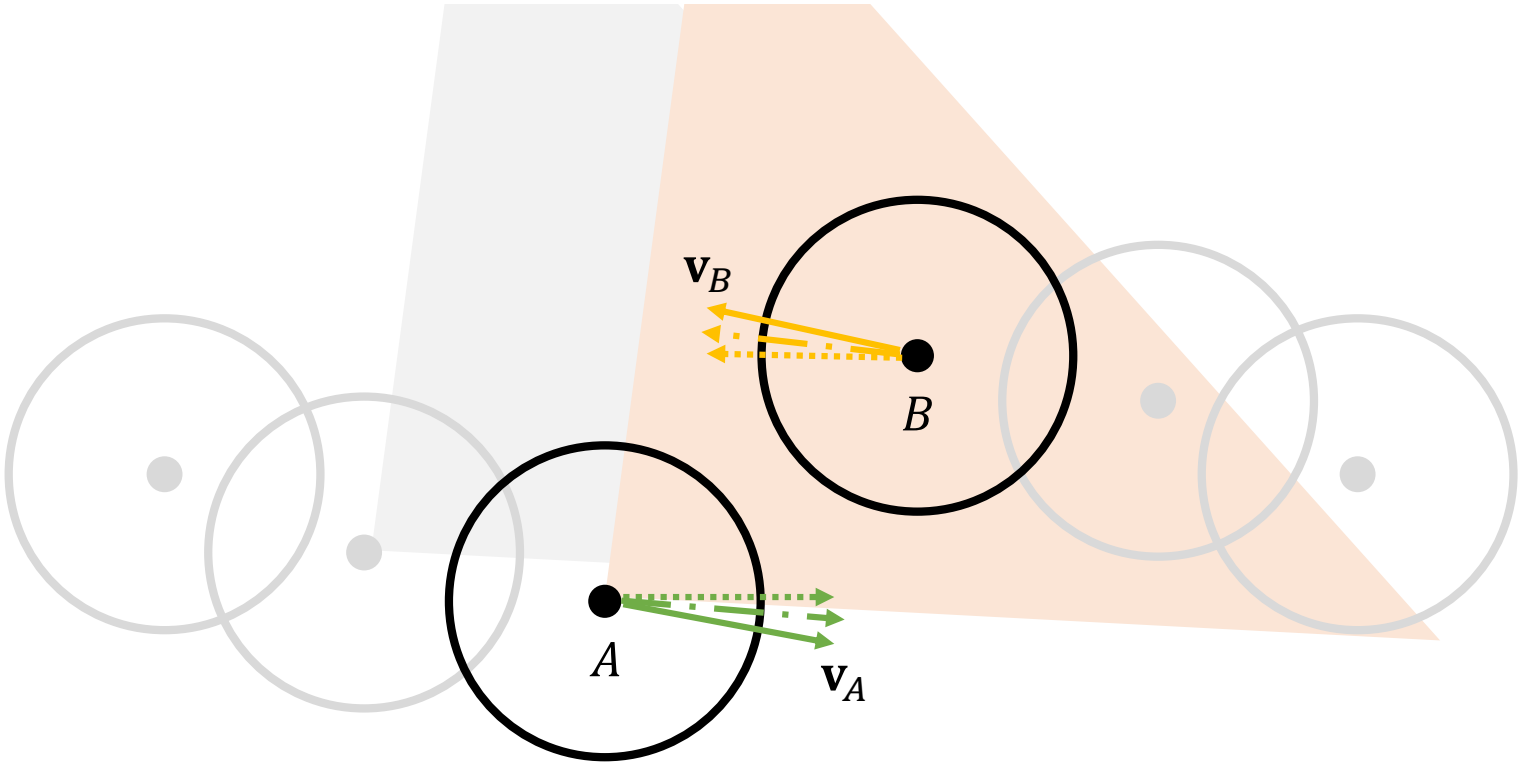
Reciprocal Velocity Obstacle (RVO)



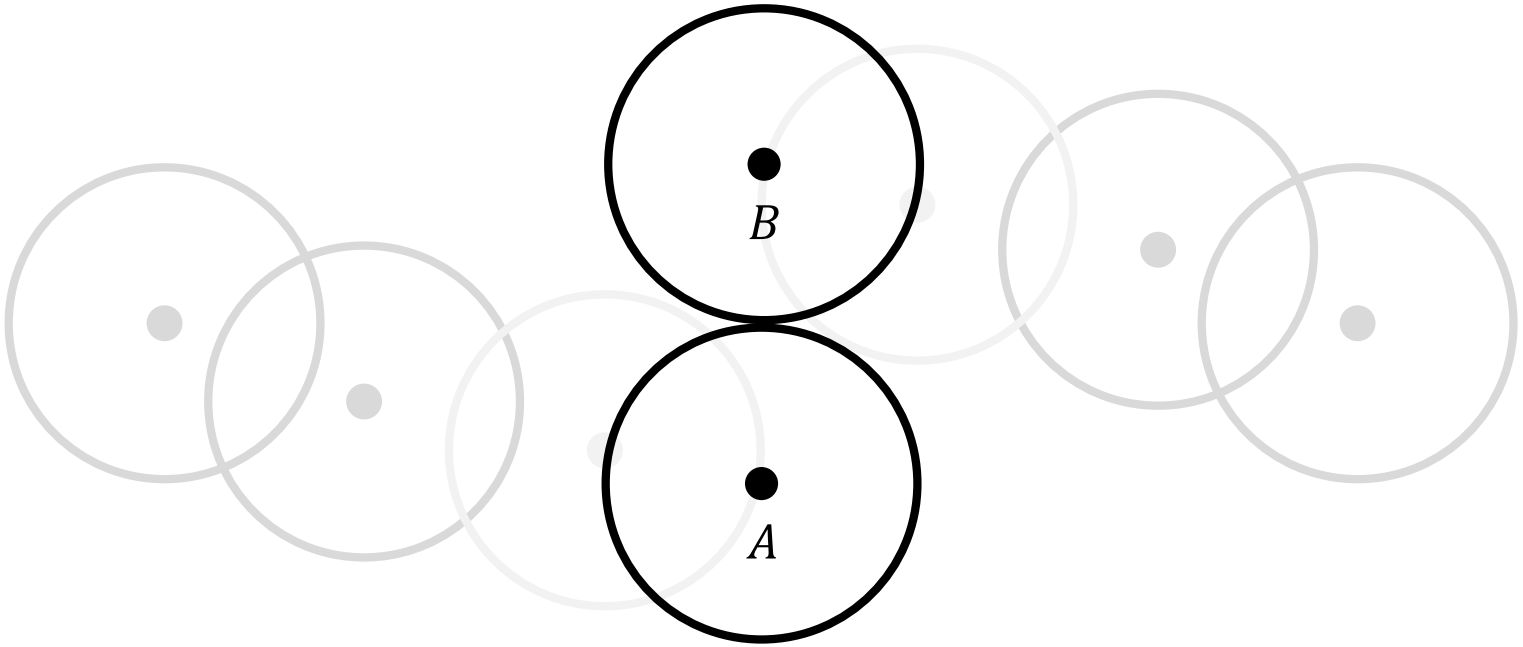
Reciprocal Velocity Obstacle (RVO)



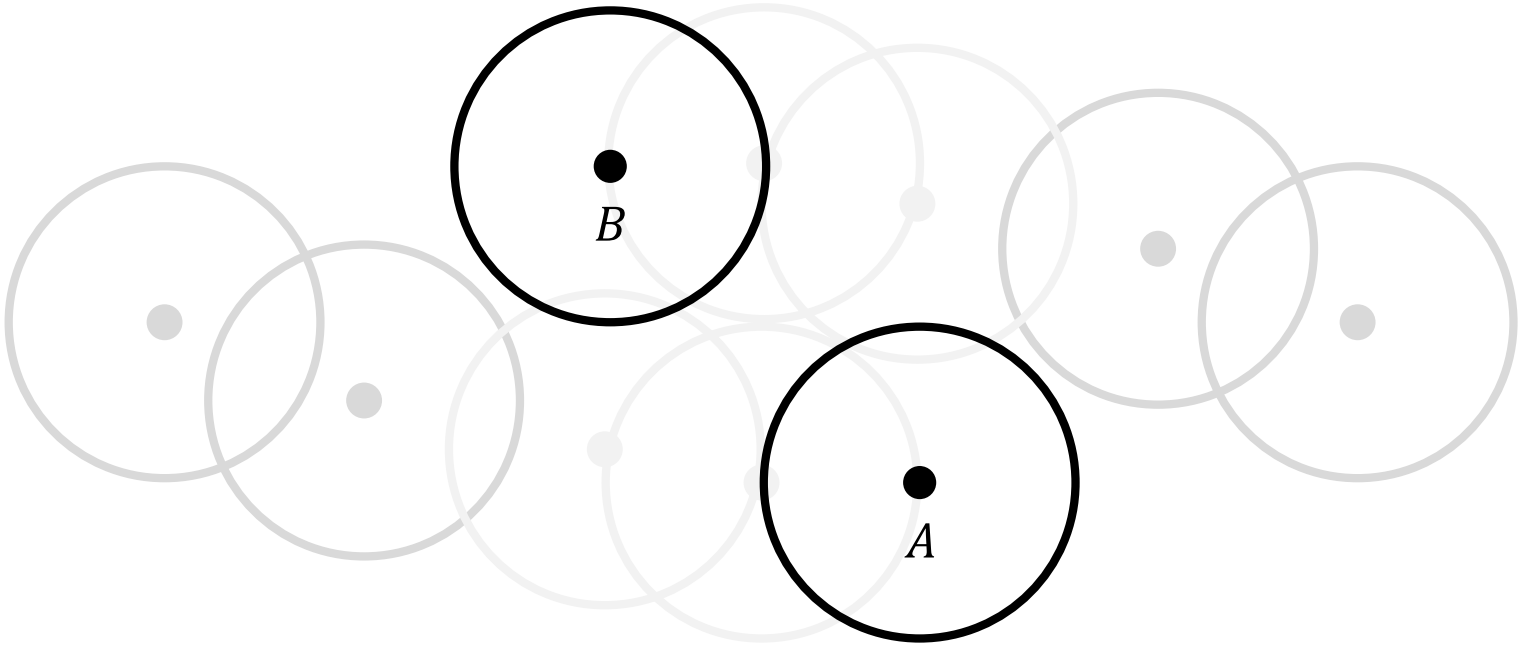
Reciprocal Velocity Obstacle (RVO)



Reciprocal Velocity Obstacle (RVO)



Reciprocal Velocity Obstacle (RVO)



Reciprocal Velocity Obstacle (RVO)

More Advanced Methods in VO

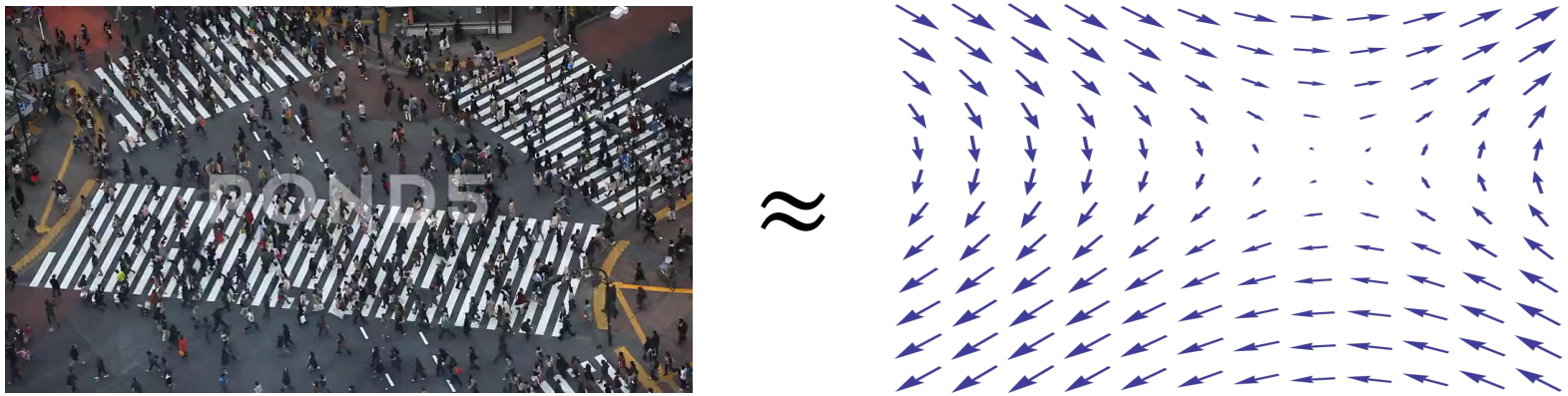
- Optimal Reciprocal Collision Avoidance (ORCA)
  - J. van den Berg, S.J. Guy, M. Lin, D. Manocha, Reciprocal n-body collision avoidance, in: C. Pradalier, R. Siegwart, G. Hirzinger (Eds.), Robotics Research, Springer Berlin Heidelberg, Berlin, Heidelberg, 2011, pp. 3–19.
- Hybrid Reciprocal Velocity Obstacle (HRVO)
  - J. Snape, J. v. d. Berg, S.J. Guy, D. Manocha, The hybrid reciprocal velocity obstacle, IEEE Trans. Robot. 27 (4) (2011) 696–706
- And many mores…

Microscopic (Agent-based) Model

- Pros
  - It resembles to what real humans/animals do
  - It can model unique behavior of each agent by simply using different rules (or parameters) for each agent
- Cons
  - Defining rules that constantly generate realistic behaviors is challenging
  - Because the defined rules only model myopic situations, global path planning quickly becomes computationally expensive, particularly in real-time contexts. In other words, huge dense crowds likely get stuck

Continuum Method (Idea)

- For crowds sharing similar goals, they almost look like continuous flows, so we can efficiently approximate those behaviors by using vector fields, from which a potential function driving people can be constructed



- The idea was first implemented by [Treuille et al. 2006]
  - A. Treuille, S. Cooper, Z. Popović, Continuum crowds, ACM Trans. Graph. 25 (3) (2006) 1160–1168

https://www.pond5.com/stock-footage/item/12431762-anonymous-crowd-moving-people-walking-commuter-commute-aeria

Hypothesis

- The overarching force driving crowd flow is that people have a destination, or ***goal***

- People ***move at the maximum speed*** possible given environmental conditions

- Even when people can move unobstructed, they may express ***preferences*** for certain paths

The Unit Cost Field

- People choose paths so as to minimize a linear combination of the following three terms:
  - The length of the path
  - The amount of time to the destination
  - The discomfort felt, per unit time, along the path

$$\underbrace{\alpha \int_P 1\, ds}_{\text{Path Length}} + \underbrace{\beta \int_P 1\, dt}_{\text{Time}} + \underbrace{\gamma \int_P g\, dt}_{\text{Discomfort}}$$

The Unit Cost Field

$$\alpha \int_P 1\,ds + \beta \int_P 1\,dt + \gamma \int_P g\,dt$$

Path Length | Time | Discomfort

$$ds = f\,dt$$

f is the field of speed
at the position $\mathbf{x}$ with the direction θ

$$\alpha \int_P 1\,ds + \beta \int_P \frac{1}{f} ds + \gamma \int_P \frac{g}{f} ds$$

Simplification

$$\int_P C\,ds \quad where \quad C := \frac{\alpha f + \beta + \gamma g}{f}$$

Cost per unit: a ***vector field*** representing cost required when moving with some direction

Optimal Path Computation

- Suppose we have a function $\phi(\mathbf{x})$: $\mathbf{R^2} \rightarrow \mathbf{R}$ gives the cost of the optimal path to the goal $\boldsymbol{g}$. Then this ***potential function*** $\phi(\mathbf{x})$ should satisfy the following conditions:

$$\phi(\boldsymbol{g}) = 0, \|\nabla\phi(\mathbf{x})\| = C$$

where the unit cost C is evaluated in the direction of the gradient $\nabla\phi$

- Then, by a theorem from the calculus of variations, it is guaranteed that all optimal paths follow exactly the gradient of this function. This means that every person moves in the direction opposite the gradient, scaled by the speed at that point:

$$\dot{\mathbf{x}} = -f(\mathbf{x}, \theta)\frac{\nabla\phi(\mathbf{x})}{\|\nabla\phi(\mathbf{x})\|}$$

Simulation Pseudocode

Initialize states

Until max-step reached

    Convert the crowd to a density field ρ

    For each group

        Construct the unit cost field C

        Construct the potential ϕ and it's gradient $\nabla\phi$

        Update the people's locations by $\dot{\mathbf{x}}$

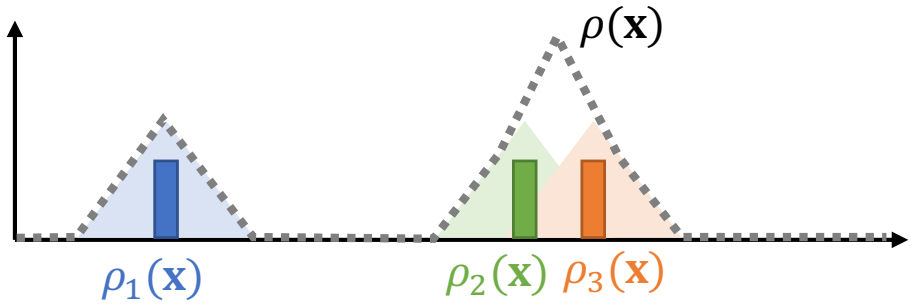
    Enforce the minimum distance between people*

*The last step is required due to discrete approximation of the fields

Continuum Method

- ***Crowd density*** field $\rho(\mathbf{x})$ refers how many people are gathered at the location $\mathbf{x}$

$$\rho(\mathbf{x}) = \sum_i \rho_i(\mathbf{x})$$



- From the density field, ***average velocity field*** $\bar{\mathbf{v}}(\mathbf{x})$ is computed, from which crowd flow can be estimated

$$\bar{\mathbf{v}}(\mathbf{x}) = \frac{\sum_i \rho_i(\mathbf{x})\,\dot{\mathbf{x}}_i}{\rho(\mathbf{x})}$$

Continuum Method

- If the crowd density is very high ($\rho > \rho_{max}$), the flow speed depends on the crowd flow only

$$f_{\bar{\mathbf{v}}}(\mathbf{x}, \theta) = \max(\bar{\mathbf{v}}(\mathbf{x} + \mathrm{r}\mathbf{n}_\theta) \cdot \mathbf{n}_\theta, 0)$$

A position that the agent would reach in near future

The facing direction
$\mathbf{n}_\theta = (\cos\theta, \sin\theta)^T$

- Meaning: it would be possible to move along with them in a similar velocity. However, the agent might not want to move in the direction of passing through them

Continuum Method

- If the crowd density is very low ($\rho < \rho_{min}$), the flow speed depends on the slope

$$f_T(\mathbf{x}, \theta) = f_{max} + \left(\frac{\nabla h(\mathbf{x}) \cdot \mathbf{n}_\theta - s_{min}}{s_{max} - s_{min}} \right) (f_{min} - f_{max})$$

$\nabla h(\mathbf{x}) \cdot \mathbf{n}_\theta$: The slope of the height field h in direction θ
s_{max}, s_{min}: The minimum and maximum slopes (constants)
f_{max}, f_{min}: The minimum and maximum speed (constants)

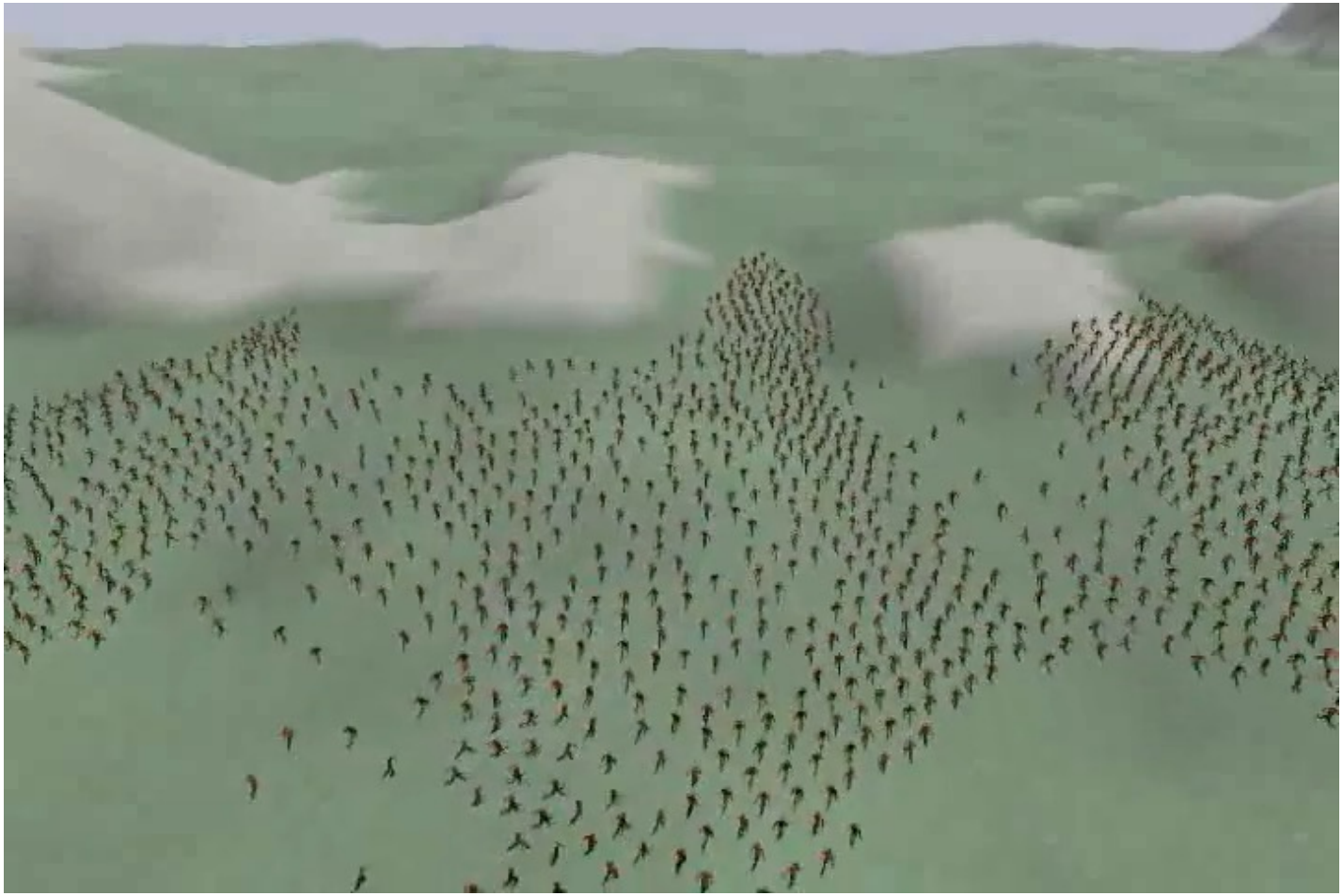
- Meaning: if the agent goes against the slope, it can only move slowly

Continuum Method

- For the crowd density ($\rho_{min} < \rho < \rho_{max}$), the flow speed is affected by both the crowd flow ahead and the slope of terrain

$$f(\mathbf{x}, \theta) = f_T(\mathbf{x}, \theta) + \left(\frac{\rho(\mathbf{x} + \mathrm{r}\mathbf{n}_\theta) - \rho_{min}}{\rho_{max} - \rho_{min}}\right)(f_{\bar{\mathbf{v}}}(\mathbf{x}, \theta) - f_T(\mathbf{x}, \theta))$$

Continuum Method



https://youtu.be/lGOvYyJ6r1c

Continuum Methods

- Pros
  - It yields a set of dynamic potential and velocity fields over the domain that guide all individual motion ***simultaneously***, in other words, global path planning and local collision avoidance are unified
  - It also guarantees that paths are ***optimal*** for the current environment state, so agents never get stuck in local minima
- Cons
  - Crowd scenarios without a definable goal such as browsing at the mall, or wandering aimlessly are not appropriate for the continuum crowd formulation
  - It is not suitable for modeling heterogenous crowds

Summary

- We have learned
  - What crowd simulation is
  - Which methods were developed to simulate crowds
    - Boids, VO-series are available in many game engines and 3D software
- Crowd simulation in a fully 3D world has not been studied much yet
- Learning-based approaches has also not been visited much yet